Teaching a Robot the Semantics of Assembly Tasks

Thiusius Rajeeth Savarimuthu[1], Anders Glent Buch[1], Christian Schlette[2], Nils Wantia[2], Jürgen Roßmann[2]
David Martínez[3], Guillem Alenyà[3], Carme Torras[3], Ales Ude[4], Bojan Nemec[4], Aljaž Kramberger[4]
Florentin Wörgötter[5], Eren Erdal Aksoy[5], Jeremie Papon[5], Simon Haller[6], Justus Piater[6], and Norbert Krüger[1]

***Abstract*—We present a three-level cognitive system in a Learning by Demonstration (LbD) context. The system allows for learning and transfer on the sensorimotor level as well as the planning level. The fundamentally different data structures associated to these two levels are connected by an efficient mid-level representation based on so called "Semantic Event Chains". We describe details of the representations and quantify the effect of the associated learning procedures for each level under different amounts of noise. Moreover, we demonstrate the performance of the overall system by three demonstrations that have been performed at a project review. The described system has a Technical Readiness Level (TRL) of 4, which in an ongoing follow–up project will be raised to TRL 6.**

***Index Terms*—Robotic assembly, Learning by Demonstration, Vision, Object Recognition**

I. Introduction

There is a significant body of work on learning (or programming) by demonstration (LbD) [1], [2], [3], [4]. It is well known that LbD–approaches face a number of challenges that grow with the ambition to transfer the taught actions to new task contexts. Such generalization requires the detection and characterization of similarities between potentially very different contexts as well as an appropriate transformation of parameters. These parameters can be of very different types depending on the representational level at which transfer is taking place.

We introduce a system that is taught assembly tasks by human demonstration, in which learning takes place in a three-level cognitive architecture with the sensorimotor level as the lowest level and a probabilistic planner as the highest level. These two levels are connected by a mid-level vision representation, which bridges from the continuous and ambiguous sensorimotor data to the planning operators defined over a discrete state space. After learning, the system is then able to plan, monitor and execute tasks, where both monitoring and executing is done by within the same representation.

At the sensorimotor level (Fig. 1, yellow area), a first problem in an LbD–framework is that the demonstrated trajectories (see Fig. 1B) as well as the forces and torques observed in the teaching process can in general not simply be replayed by the system to arrive at a succesful action. These trajectories are usually suboptimal for the specific robot embodiment, since the human performed the action in his/her own and hence different embodiment. Furthermore, the transfer of actions in general requires a given action to be performed with significant pose differences compared to the context in which the action was taught. As a consequence, trajectories might change fundamentally, and if forces and torques are important factors of the action (as, e.g., in Peg-in-Hole (PiH) actions), their optimal choice might also change with the task context. Hence, learning actions in a LbD context on the sensorimotor level presupposes a representation that predicts appropriate parameters in terms of grasp poses (Fig. 1C) and object poses (Fig. 1A) as well as trajectories (Fig. 1B), possibly with associated force/torque profiles.

Other kinds of challenges arise at the planning level, (figure Fig. 1, red area). In a complex assembly process, it is not only required to adapt trajectories to a new context, but also to plan action sequences such that they attain a given assembly goal. To this end, usually pre- and post-conditions in discrete spaces are required to compute potential outcomes of action sequences. The transfer of a task to a new context then involves the synthesis of a new action sequence as a function of these pre- and post-conditions, as the originally taught action sequence often does not apply in the new task context due to, e.g., pose differences of objects in the start configuration or workspace constraints. Moreover, often the success of an action can only be predicted with a certain likelihood, and optimal plans in terms of action sequences with high overall success likelihood should be performed. From this it becomes evident that the planning level requires a fundamentally different representation than the sensorimotor level as well as that the information that is transferred is of very different kind than on the sensorimotor level.

The community has realized the huge gap between action representations at the planning and sensorimotor levels [5], [6], [7], [8]. In this paper, we suggest a three-level representation of actions similar to that described in [6] to close that gap (see again figure (Fig. 1, beige area)). In our representation, a mid-level stage based on so called 'Semantic Event Chains' (SECs) [9] mediates between the continuous and usually rather ambiguous sensorimotor level and the planning level which operates in discrete (probabilistic) state spaces.

It is important to mention that during teaching as well as during execution, very different aspects are learned at the different representational levels. While at the sensorimotor

[1] are with the Maersk Mc-Kinney Moller Institute, University of Southern Denmark.
[2] are with Institute for Man-Machine Interaction (MMI), RWTH Aachen University, Germany.
[3] are with the Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Spain.
[4] are with the Humanoid and Cognitive Robotics Lab, Dept. of Automatics, Biocybernetics, and Robotics, Jožef Stefan Institute, Slovenia.
[5] are with the Bernstein Center for Computational Neuroscience (BCCN), Georg-August University Göttingen, Germany.
[6] are with the Institute of Computer Science, University of Innsbruck, Austria.

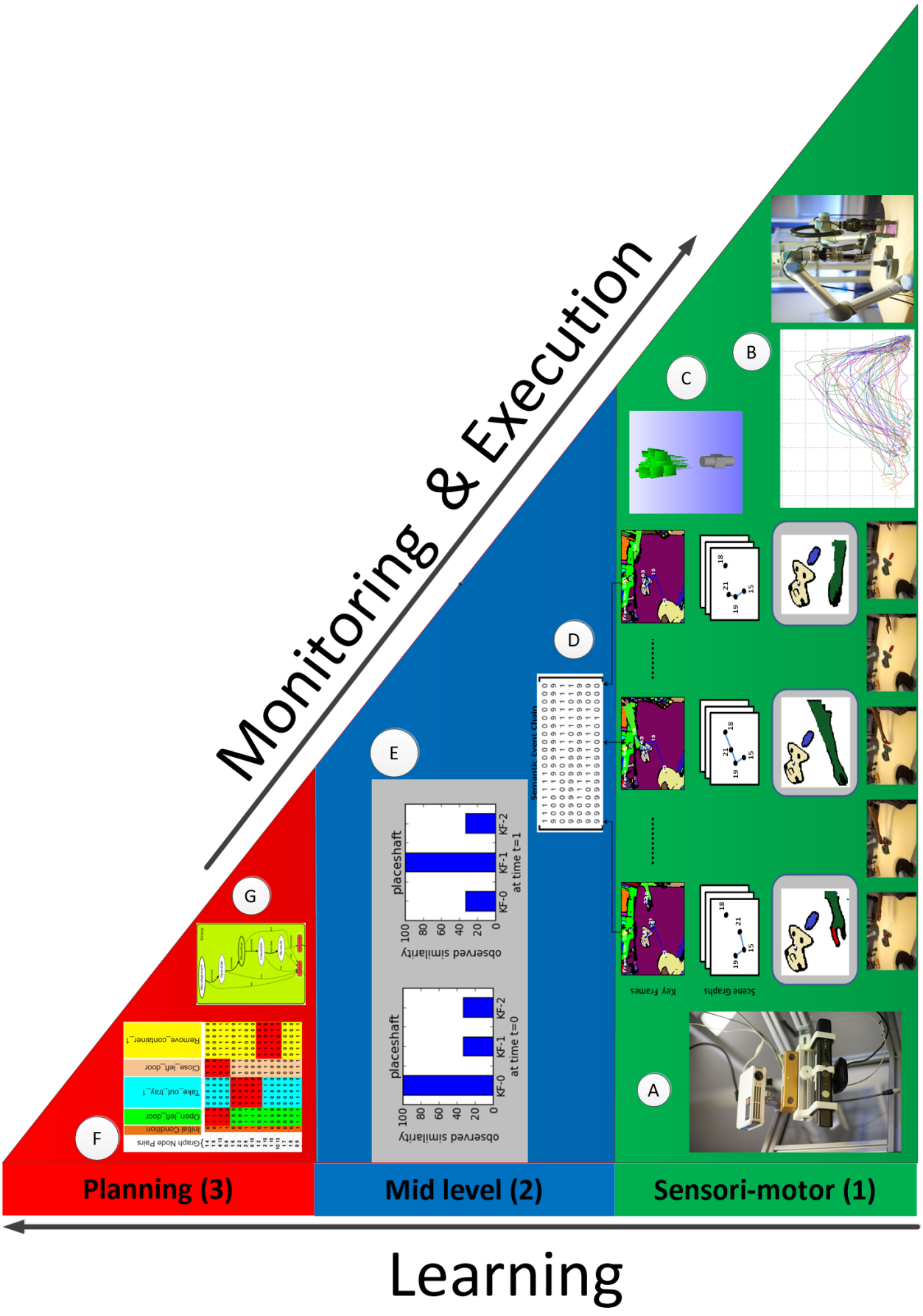


Fig. 1. Learning takes place in all 3 levels of the system: level 1) sensori-motor information is used to identify and track object movement as well as to code the force torque profile of actions. Level 2) Semantic changes in key-frames are used to identify object interaction and grouping of key-frames are used or recognize actions. level 3) Execution of actions are planned based on current observed state and possible next actions.

level the relation between grasps, trajectories, force/torque profiles and object poses are learned, at the mid-level, structural properties of actions such as relational changes between objects are acquired. At the planning level, pre- and post-conditions as well as success likelihoods of individual actions in a certain context are learned.

The different levels of our action representation can be used for transfer obeying very different purposes. At the sensorimotor level, for example, reasonable biases for potentially successful trajectories are provided. At the mid-level comparisons across potentially rather different actions can be performed with the aim of action and object substitution. At the planning level, the outcome of potentially long sequences of actions can be predicted. Consequently, complete assembly processes can be planned and executed.

We present a system in which transfer between different task contexts is taking place at all three levels. As a consequence, learning can also happen at all three levels in parallel [6].

We demonstrate the application of the three-level action representation outlined above in an industrial assembly task for which we specify the transfer at the different levels as well as outline how this transfer can be exploited for future industrial assembly systems. We address a complex assembly task, the so-called *Cranfield benchmark* (see Fig. 2). This benchmark contains a number of challenges typical for industrial assembly processes. First, the objects are of very different shapes, making it impossible to grasp each of them with only one simple gripper. Instead we use the Schunk SDH-2 dexterous hand, which is able to realize a large number of different grasp types (see Fig. 3). Secondly, insertion tasks of various kinds need to be performed in a complete assembly process. Finally, with up to 9 steps (see Fig. 2) required to perform a complete assembly process, the Cranfield assembly task exhibits a typical level of complexity for an industrial assembly setting. We want to stress that although we used the Cranfield benchmark to test and evaluate our approach, we did not make any assumptions that are specific for the Cranfield task and hence our approach can potentially be used for a wide range of assembly tasks.

This system was primarily developed during the EU-project IntellAct[1] (**Intell**igent observation and execution of **Act**ions and manipulations that was running from 2011 to 2014).

II. State of the Art

In the first sub–section, we discuss related work on the overall system level. In the subsequent sub–sections, we discuss work related to each of the three levels: In sub–section II-B, we go through related work on the sensorimotor level as well as on simulation systems. In sub–section II-C, we discuss work related to the SEC representation. Finally in sub–section II-D, we discuss related work for the planning level.

A. System Level

There are many LbD systems already applied mainly in an industrial context. It is not possible to compare our approach with all of those, hence we choose some examples to make the differences of today applied LbD systems to our approach explicit.

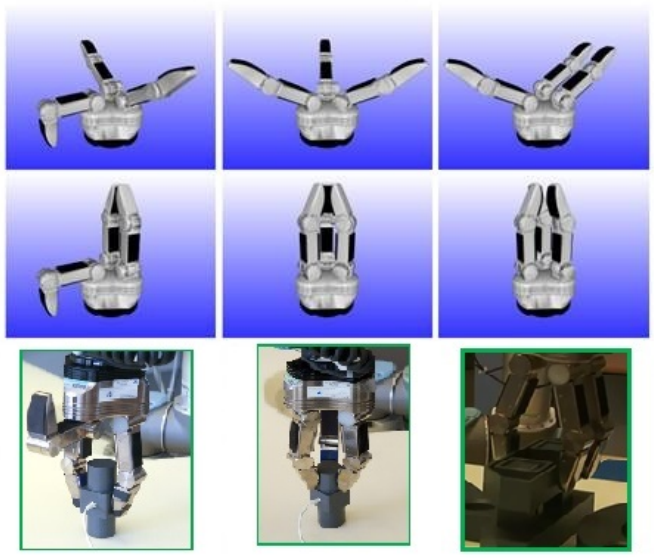


Fig. 3. Basic grasp types of the Schunk SDH-2. Left column: open and closed form state of the two-finger pinch grasp. Center column: three finger ball grasp. Right column: three finger pinch grasp.

In [10], Stopp et al. describes a manufacturing system, where the human operator and the industrial robot work together as partners in a joint manufacturing process. In this system, the operator instructs the system by specifying manipulation sequences. Each action in the sequence is programmed by means of a hand-held computer. For each action in the sequence the operator is prompted to specify the location of the object being manipulated using a laser pointer and likewise the target position of each object. The system records the information and repeats the actions as instructed autonomously to complete the assembly task. The task description and sequence is given at the start. In [11] the mobile platform, "Little Helper", is described. In this system the human operator can define tasks using either a teach- or vision-based interface. Like in the system proposed by Stopp et al., the task planning and sequencing is given by the human operator and is fixed at execution time. Lenz et al. [12] present the design of the JAHIR-demonstrator which is an assembly demonstrator allowing for joint action collaboration between human and robot in a shared workspace.

In [13], Mollard et al. presents a system using LbD to learn assembly tasks. The abstract representation of the task used for generating the action plan is constrained to interactions between two objects at a time and are extracted using physical user demonstrations. When an action plan has been generated the user can visually inspect the plan using the graphical user interface and modify it if. The system has no prior knowledge of the overall assembly goal and hence has no interaction with the user during demonstration of the assembly task as such.

The system described in this paper goes in many aspects beyond the systems mentioned above and also the systems currently applied in industry. First, the human operator demon-

[1]EU project IntellAct (FP7-ICT-269959)

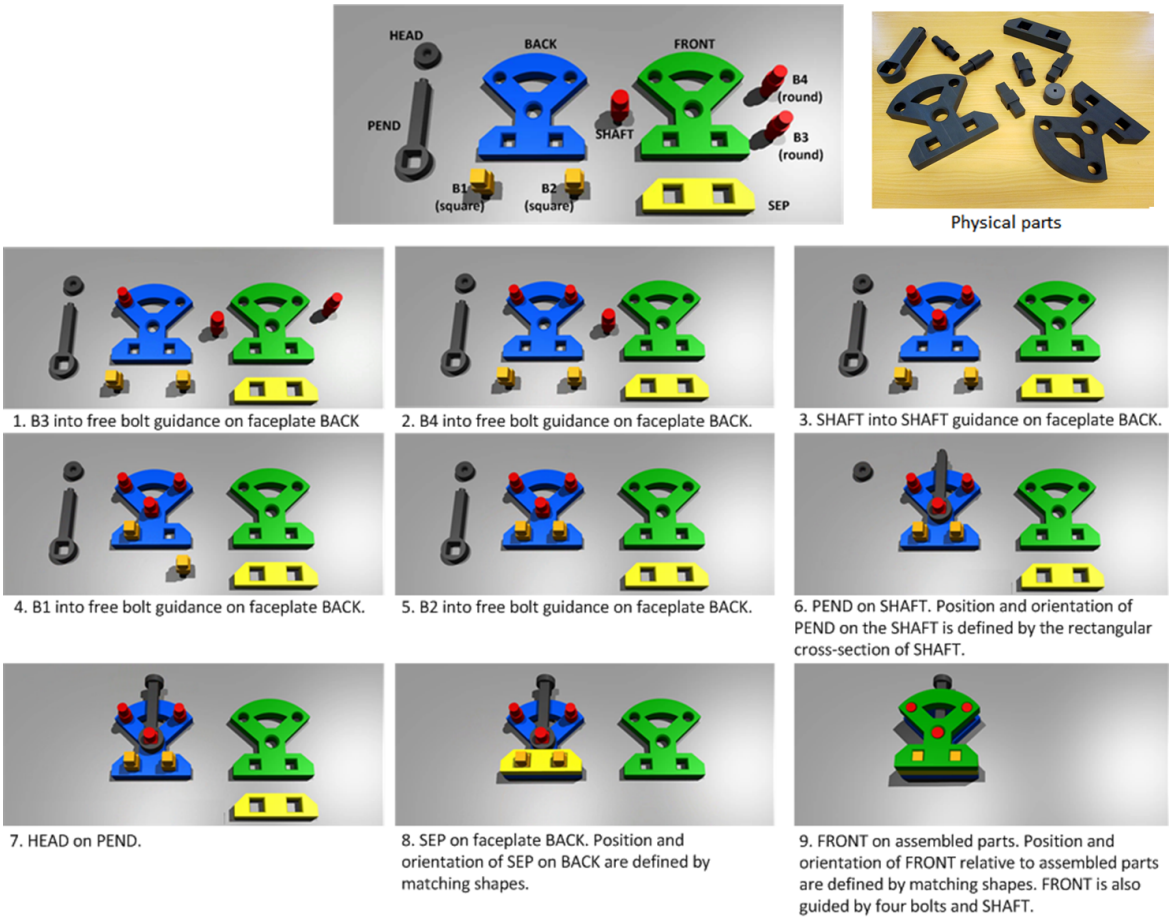


Fig. 2. One of the possible sequences of steps to assemble the Cranfield benchmark.

strates a possible sequencing of actions by performing the task using his/her own embodiment. In addition, instead of providing a manual segmentation in the demonstration process, our vision system observes the interaction between the manipulated objects and, based on this, proposes a possible sequence to complete the assembly. Compared to [13], the proposed system in this paper, has an overall assembly goal and uses this goal to evaluate the user demonstrations and the proposed system. Our system can inform the user, when the demonstrations are sufficient to reach the assembly goal. Moreover, at execution time the system generates a possible sequencing of actions based on the conditions of availability and reachability of the objects. After completing each action, the system re–evaluates the plan and continues with the execution. In this way the proposed system differs from [10] and [11] by dynamically changing the execution plan. Furthermore, the human operator can join the robotic system during the task execution by performing some of the sequenced actions. The proposed system will monitor both the robot and the human during execution and will also update the execution sequencing according to the completed actions. In the same way, the proposed system can be used to monitor task performance by human workers. Most importantly, our system is able **to learn** on three levels synchronously: It learns appropriate trajectories based on force information on the sensory-motor level, it merges action variants that are semantically equivalent into abstracted actions on the mid-level.and finally pre- and postconditions on the planning level.

B. Processing on the sensorimotor level and low-level simulation

Pose estimation and tracking: 3D object recognition and 6D pose estimation have been active research topics for several decades. An early work of [14] extracts a combination of edge and surface features for object detection. The seminal work [15] presents a recognition system based on the spin image descriptor. Mian et al. introduced in [16] a full 3D modeling and recognition system based on local descriptors called tensors. In a notable work, [17] presented a highly descriptive point cloud feature descriptor. The same descriptor was used as a plug-in feature for sophisticated recognition pipelines in [18], [19]. A recent survey [20] provides an extensive overview of current available methods for keypoint detection, feature description, and object recognition in 3D. Similar to previous works, our vision system uses a local 3D feature descriptor based pipeline for matching local structures and recovering the pose of rigid objects. Specifically, we rely on the pose estimation system [21] for fast recovery of full 6D poses.

Once initial object pose has been established, we then continuously monitor the state of each of the observed objects

using visual tracking. Multi-target visual tracking (MTVT) is a well-established field, which goes back over thirty years [22]. In this work we use the Sequential Monte Carlo method known as Particle Filtering to track targets, in particular a point-cloud 6DOF version [23] which subsamples models to work in real time. Particle filtering was first introduced to the vision community by Isard and Blake [24] and has been the subject of much subsequent research extending it [25], [26].

Robot-Control: The execution of a desired assembly task by a robot has to deal with inaccurate localization of objects in the workcell by vision and tight-tolerance operations that are common in the assembly of many products. Consequently, the robot must be compliant to successfully execute the assembly task, allowing the modification of the trained movements. Many assembly operations can be considered as a variant of a peg-in-hole task (PiH), which has been extensively investigated in the past and was also in focus of our investigations. Although many assembly problems can be solved using passive compliance, only robots with active force feedback can deal with more difficult assembly problems where larger localization errors and operations with tighter tolerances occur [27].

Active force feedback is therefore often used in robot assembly - including in our work - and regardless of whether the underlying robot is admittance- [28] or impedance-controlled [29]. However, active force control approaches usually require high feedback gains in order to adapt to the unexpected environment changes, which can cause contact instability in assembly tasks [27]. To speed up the task execution while avoiding high-gain feedback control, we propose to apply modern robot learning and adaptation approaches [30]. The basic idea of our approach is to gradually improve task execution, starting with slow task performance and increasing the speed of execution in the follow-up task executions using iterative learning control.

Exploiting Virtual Reality: Comparisons of real and virtual data have been carried out before within numerous simulation systems such as V-Rep [31], Gazebo [32] or Microsoft Robotics Developer Studio [33]. However, our approach exceeds the standard camera simulation of other simulation software as it allows for simulating various optical and electronic effects in real-time, due to utilization of rasterization techniques that can be implemented in modern shaderdriven GPUs for hardware accelerated real-time rendering [34].

Simulation is a well-established tool for the development of automated systems, but we go beyond a purely sensor- and image-based output and emulate mid- and high-level data as it is generated by a combination of sensory and processing components [35]. This allows for bootstrapping the complete system at a time when its individual components have not been implemented, yet. Furthermore, we are adding a multi-screen stereoscopic rear projection and an ultrasonic tracking system for absolute movement tracking as well as a wireless dataglove for hand movement detection for dynamic interaction with a human operator.

C. Mid-level SEC representation

In this work, Semantic Event Chains (SECs) are proposed as a mid-level action representation for assembly tasks. The main aim of employing SECs as a mid-level processor is to encode the continuous low-level signals as a sequence of descriptive symbolic states that represents the task topology. Such state sequences are indicative for the monitoring task of actions. There is a large body of work on topics related to action monitoring in computer vision and machine learning.

Considering the type of actions performed, the previous action recognition related works can be categorized in two major groups. The first group ([36], [37], [38], [39]) focuses on monitoring of full body motions, such as *walking* and *running* by considering the intrinsic hand or body movement features. The second group ([40], [41], [42], [43], [44]), on the other hand, investigates manipulation actions (e.g. *pick&place, pushing*) in which interactions between objects and hands play the most crucial role in the process of extracting the discriminative action cues. Industrial assembly tasks, as addressed in this work, fall into this type of actions.

Along these lines, the work presented in [40] introduces a method for encoding the whole manipulation sequence in a single activity graph. The main difficulty here is that very complex and large activity graphs need to be decomposed for the further recognition process. In the work of [41], segmented hand poses and velocities are used to classify manipulation actions. A histogram of gradients approach with a support vector machine classifier is used to categorize manipulated objects. Factorial conditional random fields are then employed to compute the correlation between objects and manipulations. However, this work does not consider interactions between the objects. Different from this, visual semantic graphs were proposed in [42] to recognize abstract action consequences (e.g. *Assemble*, *Transfer*) only based on changes in the structure of the main manipulated object. The work in [43] presented a method for hierarchical estimation of contact relationships (e.g. *on*, *into*) between multiple objects. The previous work [44] suggested extraction of abstract hand movements, such as *moving*, *not moving* or *tool used*, to further reason about more specific action primitives (e.g. *Reaching*, *Holding*) by employing not only hand movements but also the object information. Although all those works to a certain extent improve the recognition of manipulations and/or objects, none of them addresses the problem of deriving key events, i.e. primitives of manipulation tasks for executing the observed actions with robots.

On the other hand, high-level grammars with generative models, e.g. Hidden Markov Models (HMMs) [45], [46] and also discriminative frameworks based on multi-class Support Vector Machines (SVM) [47] and semi-Markov models [48] were proposed to reach to the level of simultaneous action segmentation and recognition. High-level grammars model the transitions between single actions in order to monitor action sequences by computing the minimum cost path through the network using efficient dynamic programming techniques. The main drawback here is the requirement of a large amount of training data to learn a state sequence and transitions for

each action. Generative and also discriminative models are generally based on bottom-up continuous movement trajectories that have high variability in appearance and shape due to differences in demonstrations performed in various scene contexts with different objects. In contrast to the aforementioned monitoring approaches, we propose a method that is based on the semantics of observed actions without being affected by the low level data variations in object or trajectory domains.

Recent works such as [49] described a Markov random field based model for decomposing and labeling the sequences of human sub-activities together with manipulated object roles. In the modeling process they employed human skeleton information, object segments and the observed object tracks. Likewise, [50] introduced a Bayesian model by using hand trajectories and hand-object interactions while segmenting and estimating observed manipulation sequences. In contrast to generative HMM - based frameworks, the SEC representation of actions also obeys the Markovian assumption. The main difference here is that all states, i.e. key frames, in the event chains represent topological changes in the scene and are fully observable. Furthermore, since detailed movement variations are not considered, event chains do not require a large corpus of training data for learning individual actions [51].

D. High-level planning system and Execution

On the highest level, we have a planning system that determines the best sequence of actions that should be executed to complete the assembly. It requires a set of planning operators, which may be handcrafted or learned. Below we provide an overview of techniques that have proven effective to learn planning operators for robotic tasks.

The main challenge in such context is to reduce the number of training actions, so that the learning phase can be completed in a reasonable time. Two techniques that allow a robot to learn fast are relational reinforcement learning (RL) and teacher demonstrations.

In relational RL, a relational representation is used to generalize the acquired knowledge over objects of the same type, which reduces greatly the number of actions required to learn [52], [53]. Lang et al. [54] have improved even further the performance with the REX algorithm, which uses count functions to apply relational generalization to the exploration-exploitation dilemma, and thus, it learns domains with very reduced amounts of exploration.

On the other hand, the ability to request demonstrations from a teacher can also speed up learning. In some approaches the teacher has to intervene to improve the robot behavior whenever it is not sufficiently satisfactory [55], [56], [57]. However, an algorithm that can actively request demonstrations when needed is preferred, as it releases the teacher from having to monitor the system continuously. Active demonstration requests have been included in algorithms with confidence thresholds [58], which request demonstrations for a specific part of the state space whenever the system is not sure about the expected behavior. A confidence-based method was also described in [59], which was combined with supplementary corrective demonstrations in error cases. Agostini et al.'s approach [60] requests demonstrations from the teacher when the planner cannot find a solution with its current set of rules.

In contrast, we use the REX-D algorithm [61], which combines relational RL and active demonstration requests. REX-D requests demonstrations only when they can save a lot of time, because teacher's time is considered to be very valuable, and uses autonomous exploration otherwise. In addition, it also applies the relational generalizations of REX [54] to yield a new algorithm that can learn with fewer action executions and demonstration requests than previous approaches.

Finally, as robot actions are not expected to be perfect and our representation of the state may lack information, the effects of actions executed by the robot will have uncertainties. A probabilistic model is learned with optimization methods in [62], but the restrictions for the initial set of candidate rules need to be manually coded. In the KWIK framework [63], a method was proposed for learning the probabilities associated with a given set of action effects using linear regression [64], as well as an extension for learning the action effects themselves [65]. However, a large number of samples are needed because the problem of learning action effects is NP complete. In our proposed method, we integrate the rule learner proposed by Pasula et al. [66] in REX-D, which employs a greedy algorithm to obtain rule sets that optimize a score function. Although this does not guarantee finding an optimal solution, it generates good rule sets based on only a few experiences. Furthermore, it generates rules with deictic references and noisy effects, which make models more compact and tractable.

III. Cranfield benchmark and MARVIN platform

To test and evaluate the system, we have created the MARVIN platform, which is a robotic platform designed to perform industrial assembly tasks (see Fig. 4). The setup includes both perception and manipulation hardware. The perception hardware includes three sets of vision sensors, each set consisting of a Bumblebee2[2] stereo camera, a Kinect sensor as well as a projector which on demand projects texture on the scene to improve stereo processing. The three camera sets are placed with approx. 120° separation, as shown in Fig. 4. In addition to the cameras, the platform is also equipped with high-precision trakSTAR magnetic trackers[3] capable of providing 6D poses simultaneously from up to four sensors which we use for teaching.

The manipulation hardware consist of two 6 degree of freedom (DOF) robots of the type UR5. At the TCP of one of the robots, a 6 DOF force-torque sensor of the type IP60[4] is mounted. Furthermore one of the robots is equipped with an SDH-2 dexterous hand.

The MARVIN platform is used to assemble the Cranfield benchmark. There are 9 steps in the assembly of the Cranfield benchmark as seen in the Fig. 2. Some of these steps are interchangeable and hence can be performed in parallel such as step 1 to 5. However step 6 can only be performed after step 3 and hence these two steps must be performed sequentially.

[2]http://www.ptgrey.com/products/bumblebee2

[3]http://www.ascension-tech.com/realtime/RTtrakSTAR.php

[4]http://www.ati-ia.com/products/ft/ft_models.aspx?id=Delta

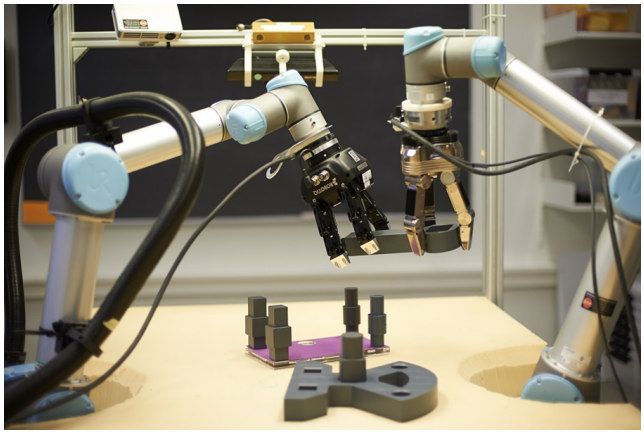


Fig. 4. The robotic MARVIN platform with two manipulators and three camera pairs.

In the same way, step 4 and 5 must be performed before step 8. Within these 9 steps there are 6 different assembly actions. These are PiH actions for round pegs (used in step 1 to 3), PiH actions for square pegs (used in step 4 and 5), the placement action of the pendulum (step 6), the screwing of these pendulum head (step 7), the placement of the separator (step 8) and finally placement of the faceplate (step 9).

IV. Virtual Testbed support for system development and optimization

Right from the beginning, the system development was accompanied by a Virtual Testbed - a 3D simulation environment to integrate, test and optimize the individual methods for learning, monitoring and execution (see Fig. 5. As a central part of the "eRobotics" methodology, Virtual Testbeds [VTBs] previously have been applied in space and field robotics to support and accelerate the development of complex technical systems [67]. In a VTB, the target system is modelled and simulated in a comprehensive 3D simulation environment which – via plugins – provides components for, e.g., programming and control of kinematics, rigid-body and sensor simulation, as well as a variety of means to connect and exchange data with other systems. The representation of a target system in a VTB allows for requirements analysis, system design and design validation based on the simulation of subsystems, the overall system, and the system in its target environment. Based on calibrated components, the level of detail of the simulation allows for the development of data processing algorithms and control schemes for operating and controlling the simulated system. The algorithms and schemes developed in the VTB are then transferred to operate and control the real system using methods of hardware/software-in-the-loop and simulation-based control [68].

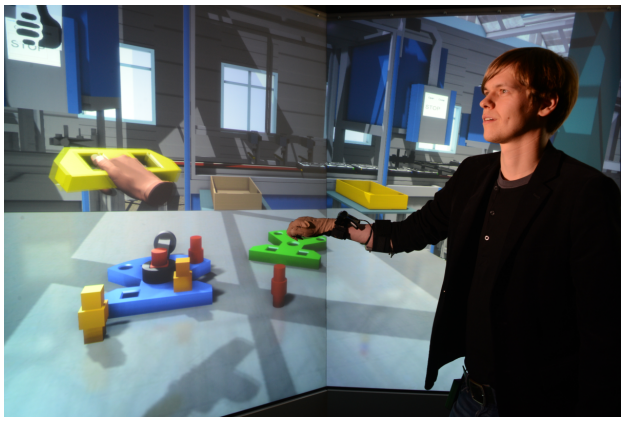


Fig. 5. Learning in a Virtual Testbed.

A major problem which often arises with complex technical systems is that functionalities of modules are too closely coupled and high-level modules depend on the availability and readiness of low-level modules close to the target hardware. Thus, efficient integration and testing is typically available only in later phases of system development.

The IntellAct project required the development and deployment of advanced software modules for observation and planning, while the target hardware and low-level modules were still in preparation. Thus, a VTB of the hardware as well as other components such as the Cranfield benchmark and certain software components was set up as a virtual substitute for the components for robot execution, sensorial output and the tracker in order to provide ideal data of objects, kinematics and sensors in several application scenarios. Only a few months into the project, the VTB served as a reference and source for ground truth data for bootstrapping the design, training and benchmarking of the high-level modules. Ground truth data was generated by carrying out object manipulations with data gloves, where object and joint positions, contact events, bounding boxes etc. were directly available from 3D simulation [69].

Beyond bootstrapping, the detailed camera and sensor simulation in the VTB allowed for offering benchmark images and point clouds with controlled levels of quality, reaching from "ideal" to "close to reality"[5]. The major advantage of generating sensorial ground truth in 3D simulation is the full transparency and control of data acquisition and the world model at each time step, thus providing otherwise unavailable details of the significant parameters. On the other hand, image generation from simulation generally faces the problem that the produced images are too ideal due to a insufficient modeling of noise and other effects. Based on results from space robotics [70], the sensor and camera simulation in the VTB supported the generation of ideal images as well as images that closely resemble the real characteristics of specific RGB and RGB-D sensor hardware. In particular, this allowed for the evaluation and optimization of the modules for pose estimation, stereo reconstruction and action recognition. This has been a problem so far, since ground truth for such algorithms is very hard to define in real setups due to the problem of estimating object poses with higher certainty than cameras would allow.

[5]Here, "close to reality" is defined by the similarity of outcomes when key factors of the real and simulated data are processed by libraries such as OpenCV and PCL, e.g. color histograms (RGB deviation, RGB saturation), edge detection, SURF feature detection and RANSAC feature similarity.

V. Recording single Actions

In the next three subsections, we will describe the three level representations, that has been sketched in the introduction, through the process of recording a single action.

A. The Sensorimotor level

At the lowest level of the proposed system, we have the sensorial and motor information. This information includes the raw motion data from the robots and grippers. Additionally, it covers the images and depth data from the vision sensors and the forces and torques data from the wrist sensor. We record these data for each assembly action in the Cranfield benchmark in a special Learning by Demonstration set–up which allows for the exploitation of the users dexterous competences (see Fig. 6). In the recording phase at the sensorimotor level, the main challenge is not of a representational kind, since basic formats are close to the signal level and hence can be defined in a straightforward way. Challenges at this level however are the stable, robust and precise visual extraction of poses and trajectories as well as an appropriate way of teaching robot actions. The recording of trajectories with associated force-torque profiles of the robot is described in subsection V-A1. Furthermore, we generate object detection and pose estimation based on the vision information acquired at this level. Pose information is then fed into a real time tracking system such that the system is able to monitor multiple objects synchronously as described in sebsection V-A2.

1) Recording motor information: There are 6 unique assembly actions in the assembly of the Cranfield benchmark - see Section III. Each of these actions is encoded in the system by human demonstration. During these demonstrations the human performs the action using the same objects as the robot and hence is able to perform the assembly task as intended with full sensorial information. At the same time the robot, in tele–operation mode, copies the movements of the human demonstrator and thereby performs the action as a copy of that performed by the human (see Fig. 6).

The trajectory and the force-torque (FT) profile of the robot movements are recorded while the human demonstrator performs the action in the teleoperation mode. During this mode, the forces and torques registered by the FT sensor in the wrist of the robot are logged along with the robot and object poses. The magnetic tracker provides 6D poses with a rate up to $200hz$. By embedding the sensors into the objects being manipulated by the human, the tracking system is able to track the movements of the object and transfer these to the robot. Fig. 6 shows a human moving a square bolt with a trackStar sensor embedded. The robot hand is holding an identical object and performing the same movements as the human. In this way, human dexterous competences can be directly transferred to the human circumventing the use of kinesthetic guidance. Kinesthetic guidance would force the teacher to work in the embodiment of the robot which prevents natural movements.

During this process, the following information is recorded and stored in a database: the initial pose of the object before manipulation begins, forces and torques measured at the robot wrist and poses of robot and the final pose of the object.

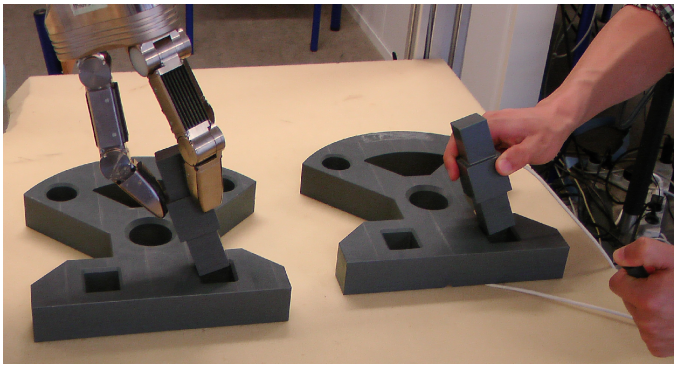


Fig. 6. Control of the robot in teleoperation mode. The robot performs a one-to-one copy of the human demonstrated path.

By recording such data for each of the six actions, we can build up an action library for the assembly of the Cranfield benchmark. Further description of the usage of the data is given in section VIII-B. These data is then used during the action execution to perform the different assembly action requested by the planner in the top level of the proposed system (Fig. 1, planning (3)).

2) The vision system: The vision system consists of two interacting modules. The first one performs object recognition and pose estimation and the other real–time tracking.

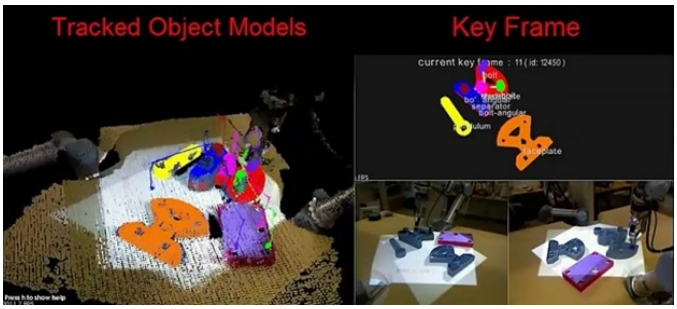


Fig. 7. The IntellAct vision system: On the left, the tracking history at an intermediate stage of the assembly is shown. The coloured threads show the positions each moved object has gone through during the assembly process. The picture on the top right shows the poses found. The two pictures on the bottom right show two images of the cameras.

Object Detection and Pose Estimation: Object Detection determines which of the Cranfield benchmark objects are present in the scene. Where they are located in the workspace is determined by pose estimation (see figure Fig. 7). To deal with this task, we represent all objects by point clouds, which can be easily generated from the available CAD models of the Cranfield objects. Then, objects are detected using a recently proposed RANSAC algorithm [21]. The 3D point cloud representation allows for finding the full 6D pose of objects, which can be immediately sent to the robot system. All poses are refined using several iterations of ICP [71] for achieving high accuracy. For this initial perception problem of locating objects, correct and accurate detections are crucial, so we use high-resolution stereo point clouds extracted from the BumbleBee cameras. This introduces a delay before any processing starts, but this task only needs to be solved once, namely at the very beginning of an assembly.

Object Tracking: Once objects and their poses are detected, the proposed system employs a 3D tracker for keeping track of all the objects in the scene in real-time. This is achieved by a novel tracking algorithm based on an octree structure, which encodes both adjacency and temporal information [72]. As an additional improvement, this structure allows for occlusion handling. If any of the objects undergoes partial occlusions during manipulation, the tracker detects that certain leaves of the octree have become occluded using a raycasting algorithm. In such cases, the leaf nodes representing the occluded parts are "frozen", and once they reappear, the tracker re-estimates the most plausible configuration of the object. This greatly increases stability of the tracker during complex manipulation sequences. The algorithm runs in near-real-time, approx. 10 Hz [73], by taking advantage of a spatially stratified sampling technique first presented in [23]. Contrary to the initial detection task, the tracker uses faster, but less accurate Kinect point cloud streams, allowing for correct tracking of all the objects at high speed.

B. Associating SECs as mid-level representation

The low level sensory information recorded as described in section Fig. V-A provides continuous streams of trajectories, poses, forces and torques. This is inappropriate for comparing actions at a semantic level, since for example very different trajectories might lead to the same topological changes in the scene. A first step required for a reasonable semantic scene interpretation is a segmentation of the continuous signal stream into meaningful chunks of discrete events that indicate unique topological changes in the scene. This segmentation and semantic condensation is achieved by Semantic Event Chains that transform the signal stream into a matrix, entries of which indicate topological changes in the scene.

'Semantic Event Chains' (SECs) were introduced in [9] as an efficient encoding scheme for manipulation actions. SECs are essentially based on consistently tracked image segments extracted from the perceived visual input stream. Each consistently segmented image is represented by a graph: nodes represent segment centers and edges indicate whether two image segments touch each other in 3D (see figure Fig. 1(A and D)). By employing an exact graph matching method, the continuous graph sequence is discretized into decisive main graphs, i.e. "*key frames*", each of which representing a topological change in the scene. All extracted main graphs form the core skeleton of the SEC, which is a matrix (see Fig. 8) where rows are spatial relations (e.g. touching) between object pairs and columns describe the scene configuration when a new *key frame* occurs. SECs consequently store sequences of changes between the spatial relations of the objects and human or robot hand in the scene. The descriptive change-patterns in SECs remain the same for a given manipulation type even when there are large variations in trajectory, pose, velocity, and objects. Thus, SECs can be used to invariantly classify manipulations as well as to categorize manipulated objects, as shown earlier in [9], [51].

Figure 8 depicts the SEC representation of a sample "Peg in Hole" (PiH) demonstration, in which a hand is first taking

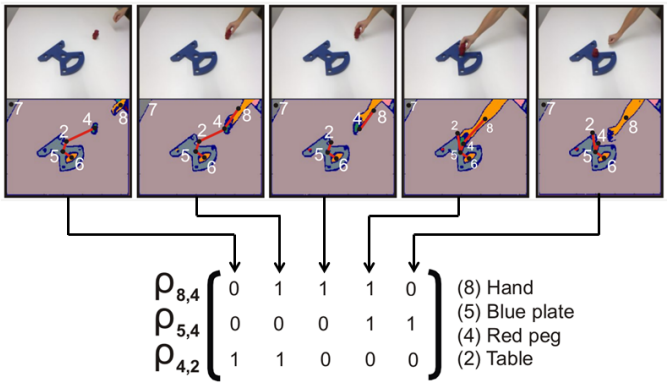


Fig. 8. Sample "Peg in Hole" (PiH) action with extracted SEC. Each SEC column corresponds to a different *key frame*. Top row shows *key frames* with consistently tracked unique segments and corresponding scene graphs. Rows describe spatial relations between objects. 1 and 0 given in the event chain stand for spatial relations touching and not-touching, respectively.

a peg and then placing it in a face plate hole. For instance, the first row of the SEC represents the spatial relations between graph nodes 8 and 4 which are the hand and red peg, respectively. Note that, although the scene involves more object segments (e.g. segment number 7), the SEC representation only encodes object pairs that produce at least one relational change from *not-touching* to *touching* or vice versa since all other pairwise relations (e.g. between the hand and table) are static and irrelevant. On top of Fig. 8, sample *key frames* including tracked segments (coloured regions) and corresponding main graphs are given to illustrate the topological configurations at the related SEC columns.

Furthermore, we associate each *key frame* in SECs with the trajectory and FT profiles of the manipulator as discussed in section Fig. V-A, since *key frames* introduce anchor points at which the continuous data can be discretized. We also enrich each graph node in SECs with respective object and pose information computed only at the decisive time points, i.e. *key frames*.

C. Association of planning operators to SECs

At the planning level, we aim to compute goal - oriented action streams by means of planning operators. Prototypical patterns of key frame sequences in a SEC can be associated to predefined planning operators representing actions such as "performPiH(objectA,objectB)" or "remove(objectA)". Moreover, symbolic states, which are extracted from SEC key frames before and after the actions, indicate pre- and postconditions of the planning operators that will be used in section VII for action monitoring and in section VIII-A for full action sequence planning.

The SEC representation is attached to three high-level modules via the Predicate Estimator and the Manipulation-Recognition modules (see Fig. 1(F)), as described in detail in sections VII-A and VII-B.

The Predicate Estimator takes each SEC keyframe, i.e. column, enriched with object poses to estimate the current state predicates as described below. The touching relations between objects are combined with the object poses to generate the

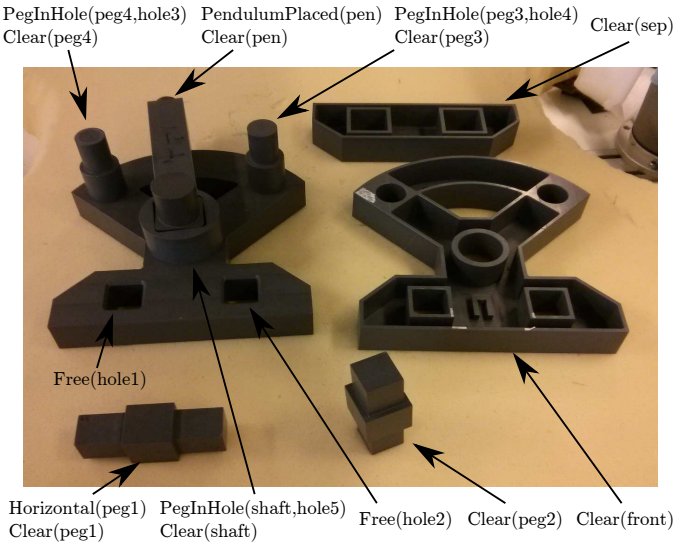


Fig. 9. An example of the state predicates used to describe the Cranfield benchmark.

predicates. SEC keyframes are passed to the Manipulation-Recognition module, which transforms them into individual actions. A set of sequential SEC columns defines a unique action, such as *peg in hole*. In VTB experiments, actions are encoded by *triplets of keyframes* corresponding to object pick-up, object transit, and object placement. However in real experiments, the number of SEC columns varies due to noise in the segmentation and tracking. Therefore actions on the MARVIN system are encoded by a *pair of keyframes* corresponding to object pick-up and object placement. Both, the state predicates and the actions are required to learn the planning operators.

A state is represented by a set of predicates that permits describing the different objects that the robot will work with. Each predicate defines a relation between two objects or a specific feature of one object. The state space consists of the following set of predicates:

- *Clear(X):* True if object X is graspable, i.e. it is in a graspable position and there are no other objects occluding it.
- *Free(X):* True if hole X is free.
- *Horizontal(X):* True if X is laying down (in a horizontal position). Pegs are much easier to grasp if they are standing up (in a vertical position).
- *PegInHole(X, Y):* True if peg X is inserted in hole Y.
- *SeparatorPlaced(X):* True if separator X has been placed.
- *PendulumPlaced(X):* True if pendulum X has been placed.
- *FacePlateFrontPlaced(X):* True if front faceplate X has been placed.

These predicates are also used to define the goal that the robot is expected to achieve. An example of a state used to describe the scenario is shown in Fig. 9. Given both the state and the goal, the planner will select the best planning operators to solve the task.

Predicates are obtained from the SEC representation enriched with object poses [74]. Whenever (see section VII-B) a new state is required, the latest SEC column is used to obtain the updated set of predicates representing the scene. Touching relations as provided by the SECs are used to identify which objects are related to each other, while poses permit checking different parts of one object. For example, if a peg is touching a faceplate, their relative positions will be checked to see if the peg is positioned in any of the faceplate holes.

Action:
PlacePeg(X,Y)
Preconditions:
peg(X), clear(X), ¬horizontal(X), hole(Y), free(Y)
Effects (Success probability: predicate changes):
0.6: ¬free(Y), PegInHole(X,Y)
0.2: ¬clear(X)
0.2:

Fig. 10. NDR rule example for placing a peg.

Planning operators are represented as Noisy Deictic Rules (NDR) [66]. Each rule r encodes the expected effects of executing an action a given a set of preconditions ϕ_{r_a}. As robot actions can fail or have unexpected outcomes, rules can have different effects that represent all the changes that an action may output and their corresponding probabilities. An example can be seen in Fig. Fig. 10. Each NDR rule refers to one action, while each action may be represented by several rules. All the rules related to the same action have disjoint preconditions $\phi_{r_{a,i}} \wedge \phi_{r_{a,j}} = \emptyset \mid \forall i, j$, so that each state-action pair (s, a) is covered by just one rule r.

Whenever an action is either executed or demonstrated, a new experience $E = [s, a, s']$, which includes the states before and after the action execution and the action itself, is stored into a library of experiences. Using these experiences, the rules that represent robot actions can be learned with a relational learner [66], where a greedy heuristic search is used since the problem of learning stochastic rule sets is NP-hard [65]. The algorithm optimizes a score function that encodes a trade-off between the accuracy and the complexity of the rules,

$$S(R) = \sum_{(s,a,s') \in E} \log \hat{P}(s'|s, a, r_{s,a}) - \alpha \sum_{r \in R} \text{PEN}(r), \quad (1)$$

where $r_{s,a}$ is the rule covering the experience when a is executed in s, $\hat{P}$ is the likelihood of the experience, $\text{PEN}(r)$ is a complexity penalty and α is a scaling parameter.

VI. Learning Action Sequences

In this section, we focus on the problem of learning full tasks, for which sequences of actions are needed to reach a goal state. Note that a large part of the training has been performed in a 3D simulation environment (see Fig. 5). In that context, action libraries with SEC models and associated planning operators are created as well as their associated trajectory and FT information. Learning on the SEC level is described in section VI-A and learning on the planning level in section VI-B. Moreover, the Decision Maker at the highest-level will decide when to request new demonstrations from the teacher, and when to execute actions on its own to learn and complete the task as outlined in section VI-B1. Note that the

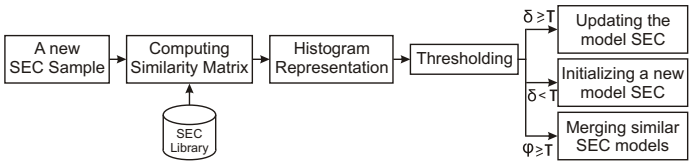


Fig. 11. Overview of the SEC learning framework.

representation at the sensorimotor level stays unchanged from what has been described for the recording of single actions. However, fine-tuning of the actual action execution results in significant speed - ups and will be described in section VIII-B.

A. Matching and generalization of SECs

An individually extracted SEC is only a sub-optimal representation of an action, because context changes and noise can lead to manifestations of rather different SECs for the same action. To be able to subsume these different manifestations into a generalized SEC representation that can also be the basis of an action library, we apply a matching and merging scheme for incoming SECs. To give evidence for the robustness of our representation at the mid-level, we investigate the stability of the SEC matching under different kinds of noise. These experiments indicate that the matching technique to compare SECs are robust to noise, in particular to the appearance of additional columns and rows in the SEC.

The main aim of the learning is to generate a library of single manipulations, e.g. *peg in hole* actions, simulated in VTB. Such a library can then be employed to monitor the observed chained actions in the real world set-up as outlined in section VII or to excute actions on the robot system (see section VIII).

Figure 11 illustrates the on-line unsupervised learning framework, introduced in [51], which is triggered whenever a new manipulation sample is observed. At start, an individual manipulation is shown in VTB and the first extracted SEC sample is assumed to be the first "model" and stored in a "SEC–library". We then encode the manipulation that follows again by a SEC and we compare it with all existing SEC models in the library. For this purpose, the framework measures semantic similarities δ between the new SEC sample and the existing models by employing the method described in [9], which compares rows and columns of two SECs using sub-string search and counting algorithms. Computed semantic similarity values between all existing models and the new sample are stored in a matrix, called the similarity matrix ζ_{sim}, which is then converted into a histogram $\mathcal{H}$ representing the distribution of similarities. We apply the conventional Otsu's method [75] to the normalized histogram in order to divide the similarity distribution into two regions representing low and high similarities, respectively. We take the average of the high similarities to estimate a threshold τ to classify the currently observed SEC sample against the existing models.

If the similarity δ is higher than τ, then the new sample will be assigned to the best fitting (most similar) model and this model will be updated with additional rows or columns that might exist in the new SEC sample [9]. In this way, the model SECs will only consist of those rows and columns observed frequently in all type-similar manipulations. If similarity δ is lower than τ, the novel SEC sample will be used as a new model in the action library. In addition, we merge learned SEC models, which have high semantic similarities, as they are likely representing the same manipulation. The merging process, in this case, searches for different rows and columns in both models and appends the novel ones to the respective model. For every new action in the library, then also a planning operator is attached as described in section V-C.

In real experiments we observed that SECs can contain not only noisy indexes (corresponding to individual digits in the SEC, see Fig. Fig. 12), but also extra noisy rows and/or columns due to noisy segmentation and tracking. Therefore, the algorithms used for analyzing SECs have to be robust against noise. In the following, we will discuss some statistical results on the robustness of our similarity measure algorithm introduced in [9].

The step of measuring the similarity between SECs plays a crucial role for the next action monitoring step. Hence, 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.

To produce more data for statistics, we first create a seed SEC that encodes a manipulation. Figure 12 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. 12. 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. The p value is then varied from 0 to 1 with a step of 0.1. Fig. 12 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 similarity method given in [9].

In Fig. 13, 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. 12. It is obvious that the slope of the curve is changing around $p = 0.5$ after which the similarity measure is around 30%. The blue curve in Fig. 13 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 mean similarity value is still around 70% even at noise level 0.8. Such high similarity values can only be observed when p is 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

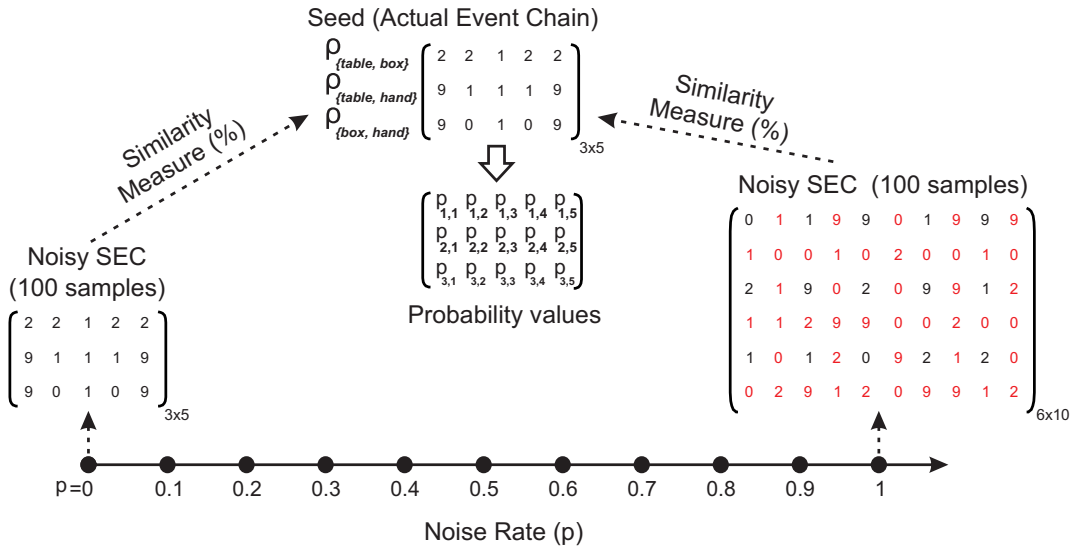


Fig. 12. *Producing noisy data for statistical analysis. For each element of the seed a probability value (p)s defined. Such probability values are also used for introducing additional noisy rows and columns. The p value is then varied from* 0 *to* 1 *with the step of* 0.1*. At each noise level* 100 *SEC samples are produced, which are then compared with the seed. 1, 0, 9 and 2 given in the event chain stand for spatial relations touching, not-touching, absence, and overlapping, respectively. Red elements in the SEC represent the noisy data, whereas those in black are the original seed elements.*

indexes remain the same. Once the SEC indexes are flipped, the similarity measure drops significantly. This is an important feature showing the importance of the original SEC indexes for the similarity measurement algorithm.

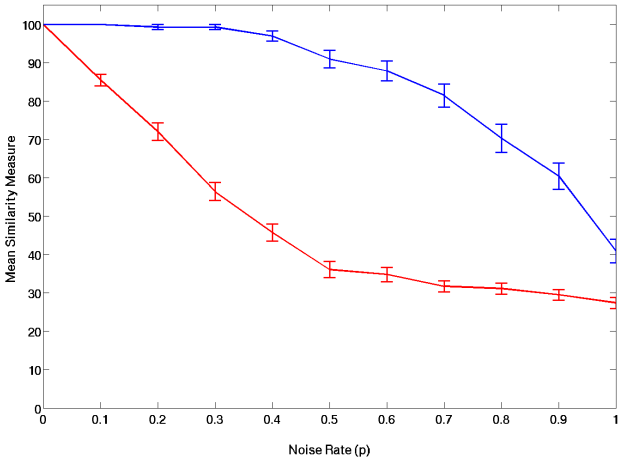


Fig. 13. *Mean values of all* 100 *similarity measures, each for one sample, at different noise levels. The red curve is for the cases when we both flip the seed indexes and add noisy rows and columns to the seed given in Fig. 12. The blue one is for the case when we add only noisy rows and columns to the same seed. The vertical bars show the standard error mean.*

Figure 13 illustrates behaviors of the similarity measures of a 3×5 seed (see Fig. 12) for two different noisy cases. Now, we would like to analyze the effects of such behaviors when sizes of SECs change. For this purpose, we created four different seeds with different sizes: 4×6, 5×7, 6×8, and 8×8. Fig. 14 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. 12.

The similarity measures between SECs can be used for classifying actions, i.e. to monitor actions. Considering the real experiments in [9], we chose a threshold at 64% that would be enough to distinguish action classes. In this regard, for further statistical analysis we can make an assumption claim that similarity between type-similar actions should be above 64% for a correct classification.

Figure 15(a) illustrates the mean similarity values with standard error means between the four seeds defined in Fig. 14 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 cannot be achieved successfully due to low similarity values.

Figure 15(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.

B. Learning on the planning level

While on the mid-level we generalize across similar actions by merging them as described above, on the planning level we learn the planning operators. For each single planning operator we learn its preconditions and effects that can be

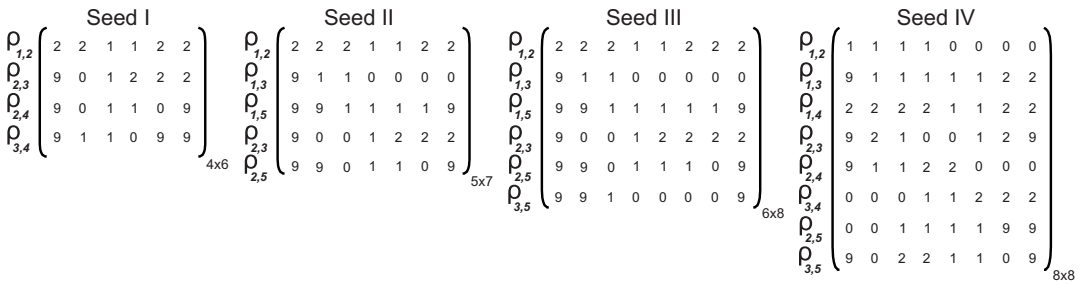
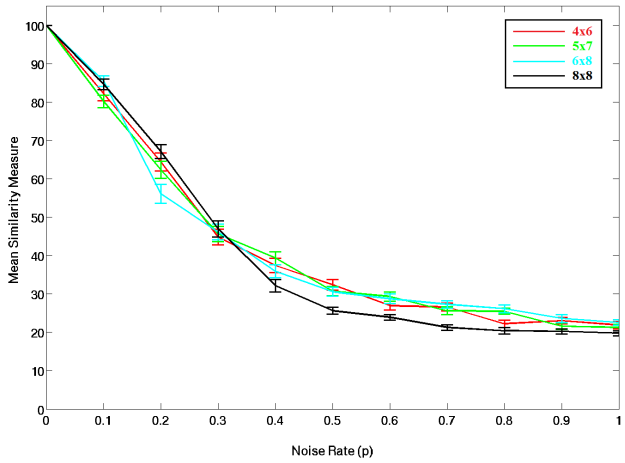
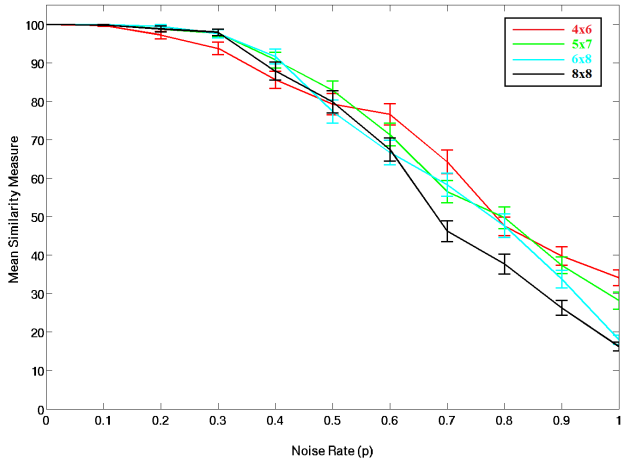


Fig. 14. *Four different seeds with different sizes:* 4×6, 5×7, 6×8, *and* 8×8. *For each seed* 100 *noisy samples are created at each noisy level by following the method illustrated inFig. 12.*



(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.*

Fig. 15. *Similarity behavior of four SEC seeds at different noise rates. The vertical bars show the standard error mean.*

computed from the states extracted from SECs as described in section V-C, as well as associated success probabilities that are estimated from a large number of input actions. These planning operators are learned in the VTB system that provides a faster and safer environment, since until the correct operators are learned, the robot may try to execute useless or even dangerous actions. Once the learning phase has been completed, the planner can be successfully integrated in the real robot system to obtain action sequences that solve the tasks and overcome possible contingencies as described in section VI-B1. As for our mid-level representation based on SECs, we also investigate the robustness of the planning operator learning under various degrees of noise in the VTB system.

We introduced the REX-D algorithm [61] to address the learning phase, which is an efficient model-based reinforcement learning (RL) method combined with additional human demonstrations upon request. It can take three alternative strategies: one is to *explore* the state space to improve the model and achieve better rewards in the long term; another is to *exploit* the available knowledge by executing the manipulations that maximize the reward with the current learned model [76]; and the last one is to request a *demonstration* from the teacher [60].

REX-D (Fig. 16) includes the exploration strategy of REX [54], which applies relational generalizations to minimize the exploration required. It explores the state space until it reaches a known state. Once in a known state, it plans using a Markov Decision Process (MDP) containing the known parts of the model, and if a plan is found, it executes it (exploitation). Note that actions may have several effects with different probabilities, and thus, a state is considered to be known when all planning operators applicable to that state have been experienced previously a number of times larger than a certain threshold. The same library of previous experiences used to learn single actions (Sec. V-C) is used to check if a state is known.

However, unlike REX, when no plan is found in a known state, instead of using planned exploration, REX-D requests a demonstration from the teacher (see section VI-B1). Actions executed or demonstrated are learned as described in Sec. V-C, adding the rule to the model so it can be used by the planner and the exploration method.

The advantage of this approach is that additional actions may be added as needed, so actions do not have to be defined at the outset. When no solution exists with the set of actions available, a teacher demonstration is requested and new manipulations can be taught. Moreover the learning time is also improved by adding just a small number of demonstrations. As the state space is usually very large, a lot of exploration may be needed, specially when there is uncertainty in the action effects

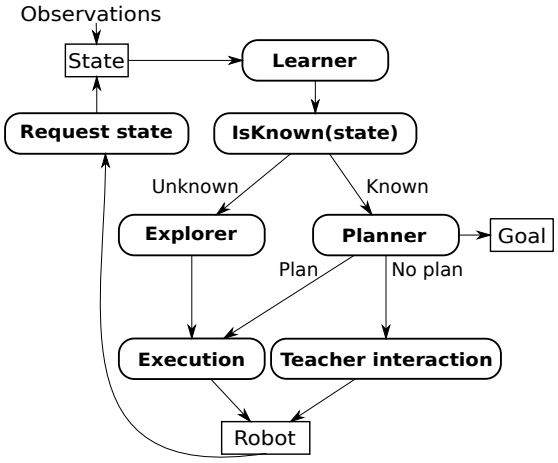


Fig. 16. Overview of the REX-D algorithm.

as the branching becomes exponential [65], but the teacher demonstrates optimal manipulations which already lead the system to those parts of the state space that will produce high rewards.

Finally, in contrast to systems with no exploration [60], REX-D maintains the number of teacher demonstrations low by adding autonomous exploration. It is preferable that the robot requests demonstrations only if they are really valuable, and explore autonomously to learn the easier parts of the domain. For example, if an action has been executed just once, there is still a lot of uncertainty about that action and executing it in different states would be very profitable to complete the model. However, when no solution can be found and all actions are already considered as known since they have been executed several times, the robot has no clues about what it should do to reach the goal, and a demonstration may save a huge amount of exploration.

1) Teacher Interaction during learning: When a demonstration is requested, several actions can be required to complete the task, and just one of them may be unknown to the system. If no guidance is provided, the teacher may demonstrate actions that the system already knows before he demonstrates the action that is actually needed. To obtain good demonstrations, the Decision Maker should inform the teacher about the reason for failure.

There are several possible reasons for a planning failure: preconditions may have been wrongly added, action effects may be missing, or a dead-end may have been reached. To determine the right explanation, we look for minimal changes in the state that would allow the planner to find a solution, which we will call *excuses* [77]. The following guidance is given to the teacher when requesting a demonstration:

- For all possible missing effects, the teacher is warned that the system does not know how to obtain the required predicates.
- For all possible wrong preconditions, the teacher is warned that an important action to reach the goal requires an unreachable precondition.

Moreover, *excuses* are also used to generate subgoals to complete all possible subtasks before requesting demonstrations. In this case, excuses permit identifying the problematic parts of the task and avoid them.

Finally, if a dead-end is reached (e.g., a piece has been broken or has fallen out of the range of the robot), the excuse will point at its cause. From that point on, whenever the planned actions may lead to a dead-end, all possible effects will be checked to ensure that the robot won't fall again into the dead-end, and request help from the teacher otherwise [78].

2) Probabilistic learning under increasing noise: The behavior of the planning system varies greatly depending on the amount of noise. As it is a common problem in robotics, actions are usually stochastic and have a chance of failure or producing unexpected effects. Therefore, in this section we analyze the performance of the decision maker with varying levels of noise in the robot actions. The REX-D algorithm is used to learn the Cranfield task in a simulated environment. Different levels of noise were introduced in the action effects to analyze the adaptability of the REX-D algorithm to uncertainty.

The results are shown in Fig. 17. As can be seen, in the deterministic case only the initial demonstrations required to learn the actions are requested, and the number of exploration actions executed is low. However, as the uncertainty in the action effects increases, the complexity to learn the scenario increases. In particular, as noise increases, extra demonstrations are requested to improve the model in uncertain cases. These few extra demonstrations allow REX-D to keep the number of exploration actions relatively low even with high levels of noise, as otherwise a huge number of exploration actions would be required until the actual model could be figured out. Moreover, note that as different action sequences can reach the goal, new unexplored states may appear in later episodes, and thus new exploration actions are triggered. Finally, the results show that the REX-D algorithm adapts very well to complex scenarios with very high levels of noise, as even in the case of 0.4 success ratio the REX-D can successfully learn good policies within a few episodes.

VII. Monitoring

So far we have described the information flow from a bottom up approach where actions are learned. In this section, we describe how the same representations are used for monitoring during action execution by a human or the robot. The monitoring system performs perception and interpretation of sensor inputs produced by both stereo and RGB-D cameras. The stereo cameras provide high-quality and high-resolution point cloud data as described in section V-A1. In order to use the system for monitoring, we need to add two modules to the system we have described so far. These are the Manipulation- Recognition module described in VII-A and the Decision Maker module described in VII-B. The Manipulation-Recognition is based on matching the perceived actions to the action library on a SEC level as described in section VI-A. Based on the state information extracted from the SECs and the associated planning operators, the Decision Maker estimates which actions can successfully complete the

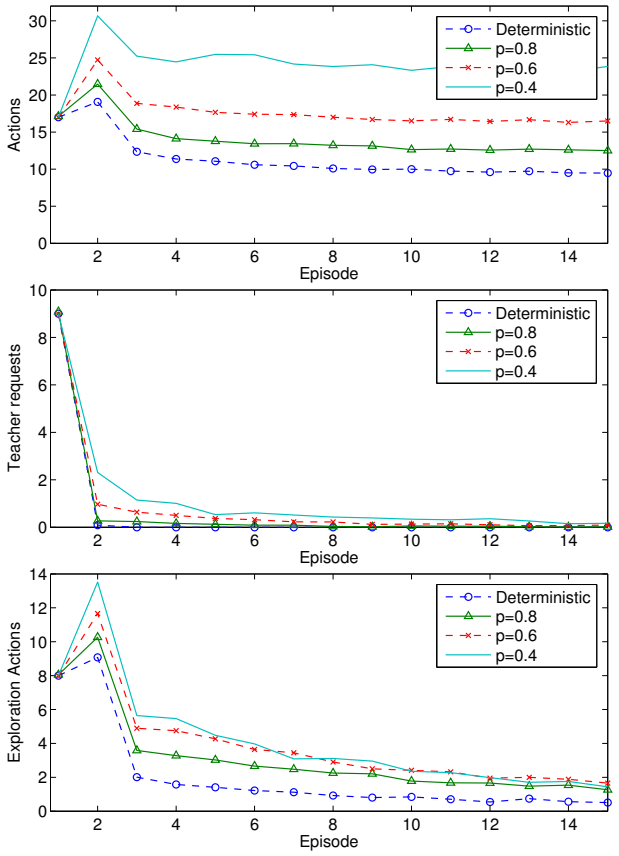


Fig. 17. The REX-D algorithm with different action success probabilities. The robot started with no knowledge of the actions in the first episode. The results shown are the means and standard deviations obtained from 100 runs. *Top:* Total number of action executions per episode. *Middle:* Number of teacher demonstration requests per episode. *Bottom:* Number of exploration actions per episode.

overall assembly task. In this way, monitoring human actions (as well as robot actions) is possible based on mid-level information encoded in SECs and high level information coded in the planning operators as described in section VII-C.

A. Manipulation recognition

The task of the Manipulation-Recognition module is to identify actions such as *shaft insertion into its hole* from the sequence of SEC keyframes, i.e. columns. It does this by matching extracted keyframes to a gallery of previously-trained keyframes that correspond to known actions. Moreover, it can recognize actions corresponding to the reversal of known actions, and can signal manipulations that do not correspond to any known action.

In principle, the Manipulation-Recognition module is capable of extracting these actions from an unsegmented stream of keyframes, by keeping track, at all times, of all possible actions in progress that are compatible with the recent history of keyframes [35]. An action is then matched and recognized in VTB by its *characteristic triple of keyframes* and on the MARVIN system by its *characteristic pair of keyframes* corresponding to object pick-up, object transit (VTB platform only), and object placement. This allows the manipulation recognition module to recognize actions by looking at the tracked state and comparing the current keyframe with the trained keyframes, considerably simplifying its operation [74].

B. The Decision Maker

The state information, extracted as described in section V-B, is fed into the Decision Maker via two distinct pathways (Fig. 1(E)). SEC keyframes are passed to the Manipulation-Recognition module, which transforms them into a sequence of individual manipulations. Secondly, object identities and poses are passed to the Predicate Estimator. Both of these two modules pass their results to the Decision Maker.

The Predicate Estimator generates the predicates required by the Decision Maker (Sect. V-C) from scene information extracted from sensor data (Sect. V-A). Most of the predicates (*Clear*, *Horizontal*, *PegInHole*, ... *Placed*) are computed from the poses of detected and/or tracked objects. The predicate *Free* employs a dedicated sensing action that checks for free space above the hole in question.

Notably, the Predicate Estimator operates in a stateless fashion. Rather than computing predicates such as *SeparatorPlaced* by remembering already-performed actions, they are computed by explicit sensing when needed. While this incurs considerable computational cost, it has the advantage that the system can detect a variety of failures and disturbances automatically, and thanks to the stateless planner can react to them without dedicated error-handling routines.

Based on these inputs and the learned planning operators (Sect. V-C), the Decision Maker computes the plan with the expected shortest distance to reach the goal. Every time an action is executed, this plan is updated to adapt to the latest changes perceived in the scenario.

C. Monitoring Human Actions

Besides closing the robotic perception-action loop by monitoring the success of robot actions, this setup allows the system to monitor *human* actions. Say, a human is to perform an assembly task. The MARVIN system has been trained and knows about the initial and final conditions and the permissible intermediate states. While the human performs the task, the system keeps recognizing and tracking objects and feeding the decision maker with state predicates and recognized actions as described above. After each individual action, the decision maker verifies that a valid plan exists, i.e., an action sequence from the current state to the goal state. If no such plan exists, it signals an error. If the shortest such plan is longer than the shortest plan prior to the latest action, it issues a warning that the user is deviating from the intended assembly sequence (see section IX-B for a description of a demo showing this).

VIII. Action Execution

In order to automatically complete an assembly task, we first need to observe the current state of the assembly sequence (as described in sections V-A and V-B), which allows the system to derive a plan that specifies which action to perform at the given state based on the learned planning operators (as described in section V-C). This computation of the next execution

step, which is based on planning and mid-level information, is described in section VIII-A. Furthermore, although the action to be performed is known at the planning level, the low-level execution of the action can be improved by learning how to perform this action in the new context. This is done by first performing a trajectory transformed to the current object pose, and then fine-tuning this trajectory iteratively to the new task context by reinforcement learning. This is done through optimizing the similarity of the FT–profile observed during teaching (as described in subsection V-A1) to the FT-profiles associated to the originally recorded trajectory. This learning process is described in subsection VIII-B.

A. Query from the planning level utilizing SEC state space information

The REX-D algorithm includes the use of an on-line probabilistic planner [79] to select the sequences of actions to complete the tasks. It requires the planning operators learned in VTB (Sec. VI-B), and the state derived from the SECs (Sec. V-C). The first action in the planned sequence will be sent to the execution modules to be performed by the robot. Once an action has finished, the state is updated and the planner generates an updated action sequence from this new state. Therefore, if something unexpected happens after executing an action, the planner will adapt afterwards and select actions that overcome the problem.

Using a planner offers a lot of flexibility to the system, as different goals can be requested without further learning. The initial state of the robot may be also changed, allowing the robot to work in other similar tasks. The used planner is probabilistic and selects the action sequence that maximizes the probability of reaching the goal. Consequently, it will take into consideration all possible effects with their associated probabilities, avoiding possible dead-ends by taking safer actions. The main limitation of probabilistic planning is that it uses computationally intensive algorithms, and tasks with large state spaces and many actions become quickly intractable.

B. Force-based learning and adaptation of sensorimotor skills

As shown in Fig. 18, we addressed both the initial acquisition of assembly skills, which in our system occurs through programming by demonstration, and later adaptation through practicing, where the initially rough skill knowledge is adapted to the kinematic and dynamic characteristics of the robot and the environment. The adaptation process occurs on the fly within the execution module, which performs actions generated by the decision maker. Skills passed to the execution module are composed of a sequence of the desired positions and orientations of the robot's tool center point (TCP) and desired tool forces and torques, expressed in Cartesian coordinates. These data are obtained as described in Section IV.A and used to compute Cartesian space Dynamic Movement Primitives (DMPs) [80]. DMPs are a suitable representation to control the robot motion. Within a DMP framework, a trajectory of every robot degree of freedom is defined by a second order linear dynamical system with an additional nonlinear term. The nonlinear term contains free parameters that can be used to adapt the movement generated by the dynamic system to the demonstrated trajectory. The desired robot positions, velocities and accelerations are obtained through integration of the equations describing the dynamical system. The major benefits of DMPs are the ability to slow down the movement via phase modulation without explicitly modifying trajectory timing and various possibilities to modulate the encoded motion, both spatially and temporary.

Assembly operations are generally subject to significant orientation changes. A nonsingular description of orientation space is provided by a unit quaternion representation. However, direct integration of unit quaternion DMPs does not preserve the unit quaternion norm. Therefore, we represented orientational motion with a specially designed dynamical system for unit quaternions, where the integration occurs directly on a manifold of unit quaternions [81], [82].

One of the major challenges of the action execution level is the robustness to the unexpected environment changes, uncertainties about the gripping pose, tolerances in the object shape, and pose estimation errors induced by the vision subsystem. Assembly skill sequences generated by the Decision Maker usually involve hard contacts with the environment, which prevents the robot from simply following the demonstrated trajectories. Hence robust execution can only be obtained by applying active force control strategies, which rely on on-line adaptation to the desired force profiles minimizing the difference between the desired and the currently measured contact forces [27]. Since the robots used in the MARVIN platform do not support torque control, we implemented an admittance stiffness PI force control law [83]. For one of the robot's degrees of freedom, e. g. y, the commanded robot position is calculated as

$$y_{\mathrm{cmd}} = y_{\mathrm{DMP}} + K_p e_f + K_i \int e_f dt, \quad e_f = f_{\mathrm{desired}} - f_y, \quad (2)$$

where y_{cmd} denotes the commanded position used to control the robot, y_{DMP} the desired position obtained from the DMP integration, f_y is the measured force and f_{desired} is the desired force obtained from human demonstration. K_p and K_i are positive scalars proportional and integral gain factors, respectively. Tuning of the integral gain K_i permits a trade–off between tracking error and stability. High gains generally result in a faster execution, but they also make force control less reliable. For example, high gains can cause jamming common in assembly tasks such as peg in hole. For this reason we used low gains for the integral term and utilized the DMP phase modulation technique to slow down the assembly task execution whenever excessive FT errors would arise. This gives sufficient time to the force controller and the robot can adapt its motion to the desired FT profile, thereby avoiding problems such as jamming. For the non-uniform scaling of the execution velocities, a DMP slowdown technique was used. More details about the DMP phase modulation approach for the Cartesian space trajectories can be found in [82].

With the proposed slowdown technique we succeeded to increase the robustness of the system, but we also extended the execution time. This drawback can be eliminated by iterative learning. Especially in industry, assembly operations

usually need to be executed many times in exactly the same configuration. In such situations humans can improve their skill knowledge by repeating the same action over and over again. The same approach is adopted by our system, where the feedback control signal from the previous repetition is reused in the current repetition of the same action. The idea is to move the force feedback error to the position displacement. The control law (2) then turns into

$$y_{\mathrm{cmd}}(l) = y_{\mathrm{DMP}} + \phi(l) + K_p e_f(l) + K_i \int e_f(l)dt, \qquad (3)$$
$$\phi(l) = \phi(l-1) + K_p e_f(l-1) + K_i \int e_f(l-1)dt,$$

where l denotes the learning cycle and $\phi(l)$ is the learned offset signal. Initially, ϕ is set to 0. More details about the learning procedure and how it is integrated into the DMP framework can be found in [30]. In this way we achieved fast and reliable execution using low integral gains. Results of learning assembly operations are shown in Fig. 18. The trajectories comprising positions and forces were captured from human demonstration. The object was then moved to a new location, which was estimated using vision. Due to the small errors in the estimated position, large force deviations arose and the algorithm slowed down the execution. After the learning was finished, the execution speed and the desired forces were close to the original demonstration time and original demonstrated forces.

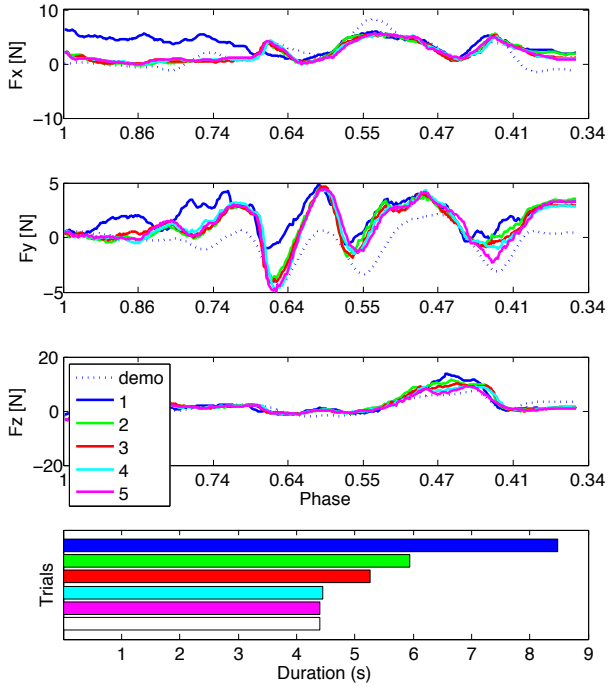


Fig. 18. The upper three graphs show the sensed forces during trajectory execution (solid lines) and the forces recorded during training (dashed lines), all as a function of phase. The bar graph below shows the execution time in each learning cycle.

IX. Demonstrations of the System as a Whole

We have given a quantitative evaluation of individual levels as it has been done in sections V, VI and VIII. However due to the high degree of complexity, it is much harder to quantitatively evaluate the system as a whole. Also it is questionable whether such an evaluation would make sense, since the system is in its way unique and possible failures can be caused by many incidents: for example some failure in the pose estimation occur due to objects being placed outside the actual workspace. Reduction of failures caused by many kinds of errors would be a primarily engineering task which was outside the scope funded in a research project.

Therefore, instead of arguing about exact percentages, we describe here the three demonstrations that have been performed at the final review of the IntellAct project and give qualitative indications about stability and frequent failures and their reasons. These demonstrations can be watched on our website in its full length http://caro.sdu.dk/index.php/videos. The first two demos show how we teach in new actions using learning by demonstration (section IX-A) as well as object detection and tracking of multiple objects (section IX-B). The final demonstration (section IX-C) shows the monitoring and execution of the complete Cranfield task on our robot platform.

A. Teaching in new actions and action fine–tuning using DMPs

The first demo shows how a single action is recorded at the sensorimotor level by LbD. The recorded action is executed in a new and random start position in the workspace. It is shown how the execution time may decrease in each execution due to the learning and optimisation of DMP parameters. In general, we were able to achieve an execution time close to the time the demonstrator required in the teaching process. Also FT unwanted peaks could be largely avoided arriving at similar forces as in the actual demonstration.

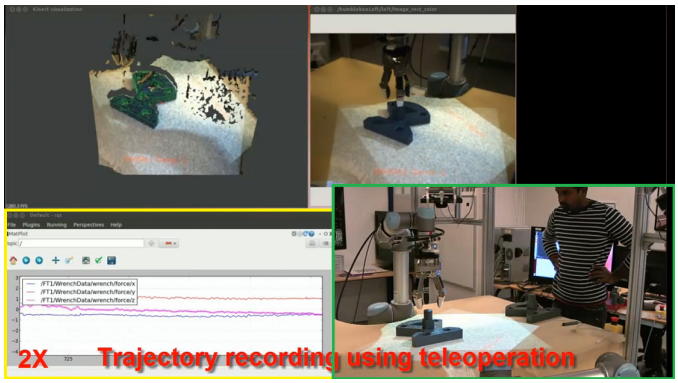


Fig. 19. The LbD demo where the user shows the system how to perform a PiH task (green window). The forces and torques acting on the peg is recorded (yellow window).

A video of the demo can be found here: https://youtu.be/c4Yc3_ES2YY and Fig. 19 displays a screenshot of the demo where the green window shows the demonstration of the task and the yellow window shows a live plot of the forces acting on the object being manipulated.

B. Object detection, pose estimation, tracking and monitoring

The second video https://youtu.be/_sRnM1e5CRY shows how object detection is used to initialize persistent object identities that are robust against occlusions and tracked throughout the entire assembly (see Fig. 20). The video shows that the system is able to track multiple objects while a human completes a subset of the Cranfield assembly task. Every assembly action is recognized by the Manipulation Recognition and the state of the system is updated according to this. The vision system performed very robustly, however only when using 2 sets of cameras. Also pose estimation for individual pegs turned out to be not robust enough due to their limited size and the noise in the point clouds. To solve this problem, the pegs were positioned in a magazine which provided enough shape information for stable pose estimation.

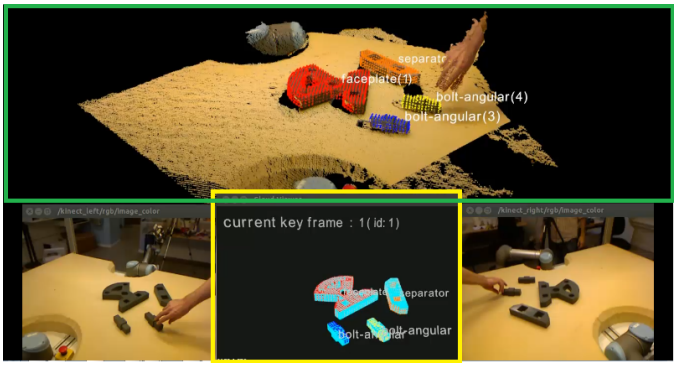


Fig. 20. The object detection and manual manipulation demo.The yellow area shows the state of the system and the current assembly state and the green area shows the live tracking of the objects.

C. Monitoring and Robot execution

In the fourth demonstration, we show the automatic assembly of the complete Cranfield benchmark (except one screwing action) as described in Sec. III. The flow of the execution is the following: when the system is activated, all the objects are in the workspace of the robot. The planner knows about all the basic assembly actions required to reach the goal state, and the sensorimotor layer gathers information of the initial state of the assembly by running object detection and pose estimation on the combined point-clouds from the 3 Kinect sensors. When the poses have been generated and the initial state of the system has been established, the planner generates a plan to reach the goal state and issues the actions to be executed.

A video of the demo can be found here: https://youtu.be/LXhzSckFy9I and Fig. 21 displays the main screen of the demo video, where the green area shows information from the mid level and the red area shows information from the top level of the system.

The execution of the complete assembly process succeeded in approximately 50% of the cases (and also at the final review only the second attempt was successful). There are various sources of failure. The action where most frequently error occurred was actually the placing of the pendulum, which usually was executed after the separator had been placed. Due

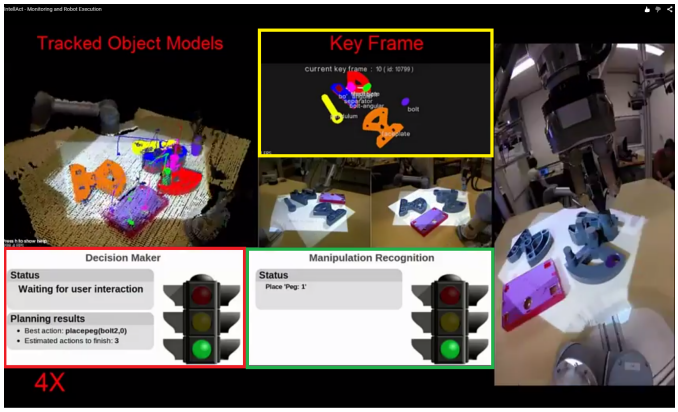


Fig. 21. The Monitoring and Robot execution video. The yellow area shows information on the sensorimotor level. The green area shows information from the mid level and the red area shows information from the top level of the system.

to the very limited space between the peg and the separator, the pose of the pendulum in the hand needed to be very accurately placed which was not always possible due uncertainties of the pose estimation but also uncertainties associated to the grasping with the SDH-2 hand as such.

X. Conclusions

We have presented a system for teaching assembly actions to robots based on a three level architecture. The system is highly flexible and is capable of monitoring both user and robotic manipulations of the objects. Learning is taking place at each level using different representations and different kinds of transfer processes. We demonstrated learning of trajectories based on force information on the sensory-motor level, matching and merging actions on the Semantic Event Chain level as well as the learning of pre- and postconditions of planning operators. All this learning can take place synchronously. We have made thorough quantifications of the learning at each level, partly making use of VR where we could introduce different noise levels.

A significant body of technologies of high complexity covering vision, planning, motor-control learning needed to be introduced and integrated to arrive at our system, that had a Technical Readiness Level (TRL) of four (validation in lab environment) at the end of the IntellAct project. In the EU project ReconCell (2015-2018), we aim to extend the system to TRL six (validation in an industrial environment). By this, the developed technology in vision, control and planning will in particular help to reduce set-up times of future robotic assembly solutions.

Acknowledgment

This work has been supported by the EU project IntellAct (FP7-ICT-269959) and the H2020 project ReconCell (H2020-FoF-680431).

References

[1] S. Schaal, “Is imitation learning the route to humanoid robots?” *Trends in Cognitive Sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[2] R. Dillmann, M. Kaiser, and A.Ude, "Aquisition of elementary robot skills from human demonstration," *Int. Symposium on Intelligent Robotic Systems, Oisa, Italy*, 1995.

[3] R. Dillmann, "Teaching and learning of robot tasks via observation of human performance," *Robotics and Autonomous Systems*, vol. 47, no. 2-3, pp. 109–116, 2004.

[4] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer Berlin Heidelberg, pp. 1371–1394.

[5] S. Coradeschi and A. Saffiotti, "An introduction to the anchoring problem," *Robotics and Autonomous Systems*, vol. 43, no. 2, pp. 85–96, 2003.

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

[7] J. Elfring, S. Van Den Dries, M. Van De Molengraft, and M. Steinbuch, "Semantic world modeling using probabilistic multiple hypothesis anchoring," *Robotics and Autonomous Systems*, vol. 61, no. 2, pp. 95–105, 2013.

[8] G. Randelli, T. M. Bonanni, L. Iocchi, and D. Nardi, "Knowledge acquisition through human–robot multimodal interaction," *Intelligent Service Robotics*, vol. 6, no. 1, pp. 19–31, 2013.

[9] E. E. Aksoy, A. Abramov, J. Dörr, K. Ning, B. Dellen, and F. Wörgötter, "Learning the semantics of object-action relations by observation," *The International Journal of Robotics Research*, vol. 30, no. 10, pp. 1229–1249, 2011.

[10] A. Stopp, S. Horstmann, S. Kristensen, and F. Lohnert, "Toward interactive learning for manufacturing assistants," *Industrial Electronics, IEEE Transactions on*, vol. 50, no. 4, pp. 705–707, Aug 2003.

[11] M. Hvilshj, S. Bøgh, O. Madsen, and M. Kristiansen, "The mobile robot "little helper": Concepts, ideas and working principles," in *Emerging Technologies Factory Automation, 2009. ETFA 2009. IEEE Conference on*, Sept 2009, pp. 1–4.

[12] C. Lenz, S. Nair, M. Rickert, A. Knoll, W. Rosel, J. Gast, A. Bannat, and F. Wallhoff, "Joint-action for humans and industrial robots for assembly tasks," in *Robot and Human Interactive Communication, 2008. RO-MAN 2008. The 17th IEEE International Symposium on*, Aug 2008, pp. 130–135.

[13] Y. Mollard, T. Munzer, A. Baisero, M. Toussaint, and M. Lopes, "Robot programming from demonstration, feedback and transfer," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems.*, 2015.

[14] F. Stein and G. Medioni, "Structural indexing: Efficient 3-d object recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 14, no. 2, pp. 125–145, 1992.

[15] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3d scenes," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 21, no. 5, pp. 433–449, 1999.

[16] A. S. Mian, M. Bennamoun, and R. Owens, "Three-dimensional model-based object recognition and segmentation in cluttered scenes," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 10, pp. 1584–1601, 2006.

[17] S. Salti, F. Tombari, and L. Di Stefano, "Shot: Unique signatures of histograms for surface and texture description," *Computer Vision and Image Understanding*, vol. 125, pp. 251–264, 2014.

[18] A. Aldoma, F. Tombari, L. Di Stefano, and M. Vincze, "A global hypotheses verification method for 3d object recognition," in *Computer Vision–ECCV*, 2012, pp. 511–524.

[19] E. Rodolà, A. Albarelli, F. Bergamasco, and A. Torsello, "A scale independent selection process for 3d object recognition in cluttered scenes," *International Journal of Computer Vision*, vol. 102, no. 1-3, pp. 129–145, 2013.

[20] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, and J. Wan, "3d object recognition in cluttered scenes with local surface features: A survey," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, no. 11, pp. 2270–2287, Nov 2014.

[21] A. Buch, D. Kraft, J.-K. Kamarainen, H. Petersen, and N. Krüger, "Pose estimation using local structure-specific shape and appearance context," in *ICRA*, 2013, pp. 2080–2087.

[22] T. Fortmann, Y. Bar-Shalom, and M. Scheffe, "Multi-target tracking using joint probabilistic data association," in *Decision and Control including the Symposium on Adaptive Processes, 1980 19th IEEE Conference on*, vol. 19, Dec 1980, pp. 807–812.

[23] J. Papon, M. Schoeler, and F. Worgotter, "Spatially stratified correspondence sampling for real-time point cloud tracking," in *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*. IEEE, 2015, pp. 124–131.

[24] M. Isard and A. Blake, "Condensationconditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[25] J. Vermaak, S. Godsill, and P. Perez, "Monte carlo filtering for multi target tracking and data association," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, no. 1, pp. 309 – 332, jan. 2005.

[26] B.-N. Vo, S. Singh, and A. Doucet, "Sequential monte carlo methods for multitarget filtering with random finite sets," *IEEE Transactions on Aerospace and Electronic Systems,*, vol. 41, no. 4, pp. 1224 – 1245, oct. 2005.

[27] W. S. Newman, M. S. Branicky, H. A. Podgurski, S. Chhatpar, L. Huang, J. Swaminathan, and H. Zhang, "Force-responsive robotic assembly of transmission components," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 3, Detroit, Michigan, 1999, pp. 2096–2102.

[28] V. Gullapalli, R. Grupen, and A. Barto, "Learning reactive admittance control," in *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, May 1992, pp. 1475–1480 vol.2.

[29] J. F. Broenink and M. L. J. Tiernego, "Peg-in-hole assembly using impedance control with a 6 dof robot," in *Simulation in Industry, Proceedings 8th European Simulation Symposium0*, 1996.

[30] F. J. Abu-Dakka, B. Nemec, J. A. Jørgensen, T. R. Savarimuthu, N. Krger, and A. Ude., "Adaptation of manipulation skills in physical contact with the environment to reference force profiles," *Autonomous Robots*, vol. 39, pp. 199–217, 2015.

[31] E. Rohmer, S. P. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 1321–1326.

[32] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3. IEEE, 2004, pp. 2149–2154.

[33] J. Jackson, "Microsoft robotics studio: A technical introduction," *Robotics & Automation Magazine, IEEE*, vol. 14, no. 4, pp. 82–87, 2007.

[34] C. Schlette, A. G. Buch, E. E. Aksoy, T. Steil, J. Papon, T. R. Savarimuthu, F. Wörgötter, N. Krüger, and J. Roßmann, "A new benchmark for pose estimation with ground truth from virtual reality," *Production Engineering*, vol. 8, no. 6, pp. 745–754, 2014.

[35] D. Martínez, G. Alenyà, P. Jiménez, C. Torras, J. Rossmann, N. Wantia, E. E. Aksoy, S. Haller, and J. Piater, "Active learning of manipulation sequences," in *Proc. of the International Conference on Robotics and Automation*, 2014, pp. 5671–5678.

[36] A. F. Bobick and J. W. Davis, "The recognition of human movement using temporal templates," *IEEE Transactions on Pattern Analysis and Machine intelligence*, vol. 23, no. 3, pp. 257–267, Mar. 2001.

[37] C. Sminchisescu, A. Kanaujia, and D. Metaxas, "Conditional models for contextual human motion recognition," *Computer Vision and Image Understanding*, vol. 104, no. 2-3, pp. 210–220, 2006.

[38] P. Scovanner, S. Ali, and M. Shah, "A 3-dimensional sift descriptor and its application to action recognition," in *Proceedings of the 15th International Conference on Multimedia*, ser. MULTIMEDIA '07, 2007, pp. 357–360.

[39] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, June 2008, pp. 1–8.

[40] M. Sridhar, G. A. Cohn, and D. Hogg, "Learning functional object-categories from a relational spatio-temporal representation," in *Proc. 18th European Conference on Artificial Intelligence*, 2008, pp. 606–610.

[41] H. Kjellström, J. Romero, and D. Kragić, "Visual object-action recognition: Inferring object affordances from human demonstration," *Computer Vision and Image Understanding*, vol. 115, no. 1, pp. 81–90, jan 2011.

[42] Y. Yang, C. Fermüller, and Y. Aloimonos, "Detection of manipulation action consequences (mac)," in *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 2563–2570.

[43] K. Nagahama, K. Yamazaki, K. Okada, and M. Inaba, "Manipulation of multiple objects in close proximity based on visual hierarchical relationships," in *IEEE International Conference on Robotics and Automation*, May 2013, pp. 1303–1310.

[44] K. Ramirez-Amaro, E.-S. Kim, J. Kim, B.-T. Zhang, M. Beetz, and G. Cheng, "Enhancing Human Action Recognition through Spatio-temporal Feature Learning and Semantic Rules," in *IEEE-RAS Int. Conf. Humanoid Robots*, October 2013.

[45] P. Peursum, H. H. Bui, S. Venkatesh, and G. A. W. West, "Human action segmentation via controlled use of missing data in hmms." in *International Conference on Pattern Recognition*, 2004, pp. 440–445.

[46] F. Lv and R. Nevatia, "Recognition and segmentation of 3-d human action using hmm and multi-class adaboost," in *European Conference on Computer Vision - Volume Part IV*, 2006, pp. 359–372.

[47] M. Hoai, Z. zhong Lan, and F. De la Torre, "Joint segmentation and classification of human actions in video," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[48] Q. Shi, L. Cheng, L. Wang, and A. Smola, "Human action segmentation and recognition using discriminative semi-markov models," *International Journal of Computer Vision*, vol. 93, no. 1, pp. 22–32, 2011.

[49] H. S. Koppula, R. Gupta, and A. Saxena, "Learning human activities and object affordances from rgb-d videos," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 951–970, 2013.

[50] A. Gupta, A. Kembhavi, and L. Davis, "Observing human-object interactions: Using spatial and functional compatibility for recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 31, no. 10, pp. 1775–1789, Oct 2009.

[51] E. E. Aksoy, M. Tamosiunaite, and F. Wörgötter, "Model-free incremental learning of the semantics of manipulation actions (in press)," *Robotics and Autonomous Systems (RAS)*, vol. 71, pp. 118–133, 2014.

[52] C. Diuk, A. Cohen, and M. L. Littman, "An object-oriented representation for efficient reinforcement learning," in *Proc. of the International Conference on Machine learning*, 2008, pp. 240–247.

[53] C. Rodrigues, P. Gérard, and C. Rouveirol, "Incremental learning of relational action models in noisy environments," in *Proc. of the International Conference on Inductive Logic Programming*. Springer, 2011, pp. 206–213.

[54] T. Lang, M. Toussaint, and K. Kersting, "Exploration in relational domains for model-based reinforcement learning," *Journal of Machine Learning Research*, vol. 13, pp. 3691–3734, 2012.

[55] Ç. Meriçli, M. Veloso, and H. L. Akın, "Multi-resolution corrective demonstration for efficient task execution and refinement," *International Journal of Social Robotics*, vol. 4, no. 4, pp. 423–435, 2012.

[56] T. J. Walsh, K. Subramanian, M. L. Littman, and C. Diuk, "Generalizing apprenticeship learning across hypothesis classes," in *Proc. of the International Conference on Machine Learning*, 2010, pp. 1119–1126.

[57] W. B. Knox and P. Stone, "Interactively shaping agents via human reinforcement: The tamer framework," in *Proc. of International Conference on Knowledge Capture*. ACM, 2009, pp. 9–16.

[58] D. H. Grollman and O. C. Jenkins, "Dogged learning for robots," in *Proc. of International Conference on Robotics and Automation*, 2007, pp. 2483–2488.

[59] S. Chernova and M. Veloso, "Interactive policy learning through confidence-based autonomy," *Journal of Artificial Intelligence Research*, vol. 34, no. 1, pp. 1–25, 2009.

[60] A. Agostini, C. Torras, and F. Wörgötter, "Integrating task planning and interactive learning for robots to work in human environments." in *Proc. of the International Joint Conference on Artificial Intelligence*, 2011, pp. 2386–2391.

[61] D. Martínez, G. Alenyà, and C. Torras, "Relational reinforcement learning with guided demonstrations," *Artificial Intelligence*, 2015.

[62] D. Sykes, D. Corapi, J. Magee, J. Kramer, A. Russo, and K. Inoue, "Learning revised models for planning in adaptive systems," in *Proceedings of the International Conference on Software Engineering*, 2013, pp. 63–71.

[63] L. Li, M. L. Littman, T. J. Walsh, and A. L. Strehl, "Knows what it knows: a framework for self-aware learning," *Machine learning*, vol. 82, no. 3, pp. 399–443, 2011.

[64] T. J. Walsh, I. Szita, C. Diuk, and M. L. Littman, "Exploring compact reinforcement-learning representations with linear regression," in *Proc. of the Conference on Uncertainty in Artificial Intelligence*, 2009, pp. 591–598.

[65] T. Walsh, "Efficient learning of relational models for sequential decision making," Ph.D. dissertation, Rutgers, The State University of New Jersey, 2010.

[66] H. M. Pasula, L. S. Zettlemoyer, and L. P. Kaelbling, "Learning symbolic models of stochastic domains," *Journal of Artificial Intelligence Research*, vol. 29, no. 1, pp. 309–352, 2007.

[67] J. Roßmann, E. Guiffo Kaigom, L. Atorf, M. Rast, G. Grinshpun, and C. Schlette, "Mental models for intelligent systems: eRobotics enables new approaches to simulation-based AI," *KI-Künstliche Intelligenz*, vol. 28, no. 2, pp. 101–110, 2014.

[68] M. Schluse, C. Schlette, R. Waspe, and J. Roßmann, "Advanced 3d simulation technology for eRobotics," in *International Conference on Developments in eSystems Engineering (DeSE)*, 2013, pp. 151–156.

[69] J. Roßmann, C. Schlette, and N. Wantia, "Virtual Reality in the loop providing an interface for an intelligent rule learning and planning system," in *Workshop on Semantics, Identification and Control of Robot-Human-Environment Interaction at International Conference on Robotics and Automation (ICRA)*, 2013, pp. 60–65.

[70] J. Roßmann, T. Steil, and M. Springer, "Validating the camera and light simulation of a virtual space robotics testbed by means of physical mockup data," in *International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, 2012, pp. 1–6.

[71] P. Besl and N. D. McKay, "A method for registration of 3-d shapes," *PAMI*, vol. 14, no. 2, pp. 239–256, 1992.

[72] J. Papon, T. Kulvicius, E. Aksoy, and F. Worgotter, "Point cloud video object segmentation using a persistent supervoxel world-model," in *IROS*, 2013, pp. 3712–3718.

[73] T. R. Savarimuthu, J. Papon, A. G. Buch, E. E. Aksoy, W. Mustafa, F. Wörgötter, and N. Krüger, "An online vision system for understanding complex assembly tasks," in *International Conference on Computer Vision Theory and Applications (VISAPP)*.

[74] T. R. Savarimuthu, A. G. Buch, Y. Yang, W. Mustafa, S. Haller, J. Papon, D. Martınez, and E. E. Aksoy, "Manipulation monitoring and robot intervention in complex manipulation sequences," *Workshop on Robotic Monitoring (at the Robotics: Science and Systems conference)*, 2014.

[75] N. Otsu, "A Threshold Selection Method from Gray-level Histograms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[76] R. I. Brafman and M. Tennenholtz, "R-max-a general polynomial time algorithm for near-optimal reinforcement learning," *Journal of Machine Learning Research*, vol. 3, pp. 213–231, 2003.

[77] M. Göbelbecker, T. Keller, P. Eyerich, M. Brenner, and B. Nebel, "Coming up with good excuses: What to do when no plan can be found." in *Proc. of the International Conference on Automated Planning and Scheduling*, 2010, pp. 81–88.

[78] D. Martínez, G. Alenyà, and C. Torras, "Safe robot execution in model-based reinforcement learning," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015.

[79] A. Kolobov, Mausam, and D. S. Weld, "Lrtdp versus uct for online probabilistic planning," in *Proc. of the AAAI Conference on Artificial Intelligence*, 2012.

[80] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical Movement Primitives: Learning attractor models for motor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013.

[81] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal, "Online movement adaptation based on previous sensor experiences," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, California, 2011, pp. 365–371.

[82] A. Ude, B. Nemec, T. Petrič, and J. Morimoto, "Orientation in cartesian space dynamic movement primitives," in *International Conference on Robotics and Automation (ICRA)*, Hong Kong, 2014, pp. 2997–3004.

[83] L. Villani and J. De Schutter, "Force control," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Berlin, Heidelberg: Springer, 2008, pp. 161–185.