Manipulation Monitoring and Robot Intervention in Complex Manipulation Sequences

Thiusius R. Savarimuthu[1], Anders G. Buch[1], Yang Yang[1], Wail Mustafa[1], Simon Haller[2], Jeremie Papon[3], David Martínez[4], Eren E. Aksoy[3]

I. Introduction

Compared to machines, humans are intelligent and dexterous; they are indispensable for many complex tasks in areas such as flexible manufacturing or scientific experimentation. However, they are also subject to fatigue and inattention, which may cause errors. This motivates automated monitoring systems that verify the correct execution of manipulation sequences. To be practical, such a monitoring system should not require laborious programming.

To this end, we recently presented such a system (the so-called Decision Maker, DM) that uses a probabilistic planner and learns rules describing manipulation sequences from previous executions and human demonstrations [1]. Operating in a virtual-reality environment, this system recognizes atomic manipulations, defined in terms of touching events between objects and extracts rules consisting of preconditions, manipulation, and effects. The rule base is incrementally extended and refined as new observations occur. Once a sufficient rule base has been acquired, the DM can monitor ongoing manipulation sequences by verifying that at each step a shorter plan towards the desired goal state exists.

In this work, we describe our physical implementation of a robot manipulation monitoring system. Figure 1 illustrates its overall architecture. It consists of four levels of functional modules. The first low-level (yellow modules) operates on sensorial data and generates information about *Tracked Objects* and their respective *Poses*. The mid-level (green modules) includes the components used to detect touching events between objects in the scene. The planning modules form the high-level (blue modules) and are used to recognize and plan actions. The final motor control-level holds the execution modules (red modules) which are responsible for learning, storing and executing robot movements. Next, each perception and execution module will be briefly described.

II. Perception

A. Low-level

1) Sensor Buffer and Tracker: Our sensor buffer collects data from various sources, and performs temporal synchronization before fusing the data into a voxelized octree world model. This is input to the tracker, which uses the object detection to bootstrap object models. Tracking is performed using a bank of independent parallel particle filters [2]. Independently tracked results are combined and interactions between objects are resolved jointly using a task-specific model.

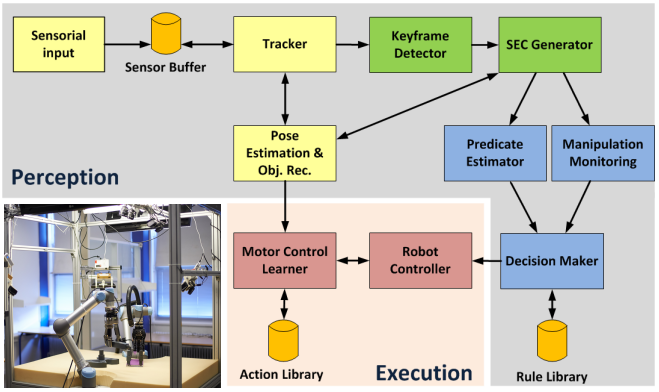

Fig. 1. System Architecture showing the interaction between the perception and execution modules. Yellow, green and blue modules represent the three levels in the perception and the red blocks are the execution modules.

2) Object Recognition: Object instance recognition is used in the very first frame produced by the sensor buffer to associate a known object identity to each of the objects placed on the workspace table. We use Euclidean clustering to separate the objects on the table, and then perform pose estimation using an optimized RANSAC routine [3]. The best matching candidate among the known objects is assigned to the perceived object cluster.

B. Mid-level

1) Keyframe Detector: We represent the entire scene with a graph: nodes represent the object centers in the scene, and edges are for the touching relations between objects. We employ extracted 3D scene graphs in order to detect topological changes in the scene, i.e. *key frames*, by comparing the eigenvalues of graphs at each frame.

2) SEC Generator: Detected *key frames* are encoded in a matrix format, so called Semantic Event Chain (SEC) [4]. The SEC generator module enriches each detected *key frame* with fine pose information imported from the low level perception. The mid-level modules are particularly providing the temporal anchor points indicating when to update the scene graphs with low level pose information, instead of recording the continuous pose data.

*The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 269959, IntellAct.

[1]The Mærsk Mc-Kinney Møller Institute, Univ. of Southern Denmark
[2]Institute of Computer Science, University of Innsbruck, Austria
[3]Institute for Physics 3, Georg-August-University, Göttingen, Germany
[4]Institut de Robòtica i Informàtica Industrial (CSIC-UPC), Spain

C. High-level

1) Predicate Estimator: The predicate estimator module evaluates the current state predicates on requests from the DM. It estimates a set of predefined predicates using the SEC representation enriched with the object poses. For example, the predicate `free(hole)` is set true if this particular hole of the Cranfield faceplate is not occupied, and `placed(object)` is defined as true in the case of having an object placed in a designated position on the faceplate at the recently detected *key frame*.

2) Manipulation Recognition: A manipulation is defined as a sequence of *key frames*, i.e. relational changes between objects. This module compares the current SEC with a set of known touching events corresponding to a manipulation such as *place peg into faceplate*. If a known touching relation is observed in the current *key frame* the respective manipulation is detected and signaled to the DM.

3) Decision Maker (DM): The decision maker learns a set of symbolic rules that define the behavior of demonstrated manipulations. Those rules are further employed either to execute the learned task even in a novel scene context, or to evaluate the robot's own performance while executing a given task. In this regard, the estimated predicates are fused with those learned rules to plan the remaining action steps to reach the ultimate goal. The learning phase is addressed using the REX-D algorithm [1] which is an efficient reinforcement learning (RL) method combined with additional human demonstrations upon request.

III. Execution

A. Motor Control Learner

In our framework the robot movements are programmed by teleoperation [5]. The teleoperation is enabled by registering 6D magnetic pose tracker to the robot and placing a pose sensor on the operator to track the demonstrated movements. During the demonstration the respective poses and associated forces are recorded and stored. The tracked movements are then employed to control the robot to imitate a particular manipulation, such as *place peg into faceplate*.

B. Robot Controller

During the execution, the Robot Controller module loads the recorded robot motion from the stored data (action library). This loaded motion is then transformed from the recorded target position to the new and actual target position observed in the scene. The transformed robot motion is finally used to perform the action.

IV. Results and Conclusion

The proposed robot execution and online monitoring framework was tested on the Cranfield benchmark objects described in [5]. Figure 2 illustrates a sample trajectory between two consecutive *key frames* detected from the robot execution of the action *place(shaft)*. At the *key frame t*, the plan *place(shaft)* was issued by the Decision Maker based on the predicate evaluation. Once the given plan is executed (red block), the performed robot action is correctly monitored at

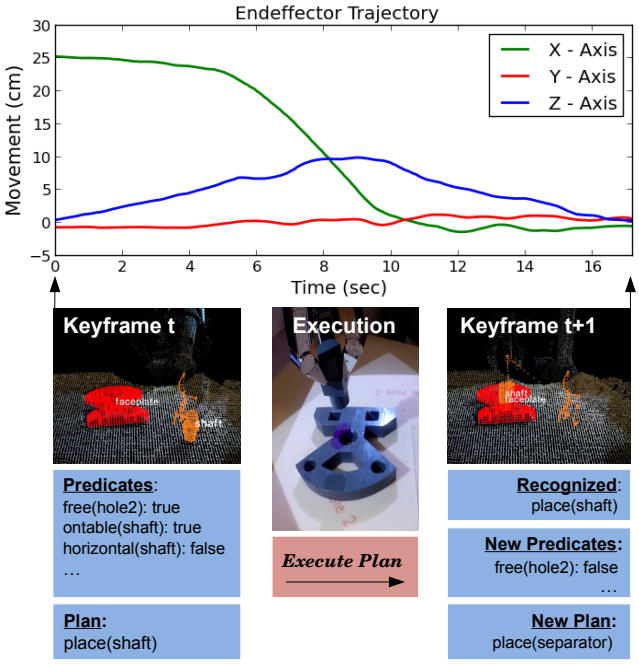

Fig. 2. Monitoring a *place(shaft)* action. **Top:** Trajectory of the robot movement. **Middle:** Key frames with tracked objects before and after the manipulation. **Bottom:** High-level planing and monitoring results.

key frame t+1 and the next plan is further estimated from the new predicates given on the right.

The evaluation experiments show that the system is able to recognize and monitor robot manipulations on the fly. We were able to distinguish and track all 9 Cranfield objects during both human and robot manipulations. The low level pose estimation is employed only once to bootstrap the tracker in the beginning of the assembly sequence. The concept of event chains discretizes the continuous motions into temporal anchor points, i.e. *key frames*, at which the pose information is further updated to decrease the computational complexity of the entire system. The complete assembly of 9 Cranfield objects by robot is shown in this video: http://youtu.be/LXhzSckFy9I

References

[1] D. Martínez, G. Alenyà, P. Jiménez, C. Torras, J. Roßmann, N. Wantia, E. Aksoy, S. Haller, and J. Piater, "Active Learning of Manipulation Sequences," in *Inter. Conf. on Robotics and Automation (ICRA)*, 2014.

[2] J. Papon, T. Kulvicius, E. E. Aksoy, and F. Wörgötter, "Point cloud video object segmentation using a persistent supervoxel world-model," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013.

[3] A. G. Buch, D. Kraft, J.-K. Kamarainen, H. G. Petersen, and N. Kruger, "Pose estimation using local structure-specific shape and appearance context," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 2080–2087.

[4] E. E. Aksoy, A. Abramov, J. Dörr, N. Kejun, B. Dellen, and F. Wörgötter, "Learning the semantics of object-action relations by observation," *The International Journal of Robotics Research September*, vol. 30, pp. 1229–1249, 2011.

[5] T. Savarimuthu, D. Liljekrans, L.-P. Ellekilde, A. Ude, B. Nemec, and N. Kruger, "Analysis of human peg-in-hole executions in a robotic embodiment using uncertain grasps," in *Robot Motion and Control (RoMoCo), 2013 9th Workshop on*, July 2013, pp. 233–239.