Spatial Constraint identification of parts in SE3 for action optimization

Jimmy Alison Jørgensen, Nadezda Rukavishnikova, Norbert Kruger and Henrik Gordon Petersen

The Maersk Mc-Kinney Moller Institute, Faculty of Engineering
University of Southern Denmark
5230 Odense M, Denmark
{jimali,nadezda,norbert,hgp}@mmmi.sdu.dk

***Abstract*— In this paper we present a method to structure contextual knowledge in spatial regions/manifolds that may be used in action selection for industrial robotic systems. The contextual knowledge is build on relatively few prior task executions and it may be derived from either teleoperation or previous action executions. We argue that our contextual representation is able to improve the execution speed of individual actions and demonstrate this on a specific time-consuming action of object detection and pose estimation.**

Our contextual knowledge representation is especially suited for industrial environments where repetitive tasks such as bin- and belt picking are plentiful. We present how we classify and detect the contextual information from prior task executions and demonstrate the performance gain on a real industrial pick-and-place problem.

I. Introduction

In industrial automation there is an increasing demand for fast reconfigurable robot platforms. Currently, reconfiguring a robot platform requires time consuming and expensive manual labor, often by expert roboticists. The main reason is that industrial programs tend to be very specific, requiring the offline definition of most spatial knowledge in the workspace. This could for example be object poses from which camera views and grasping motions are defined. There are two drawbacks of such labor-intensive program definition: (i) lost flexibility and (ii) experts are needed for any reconfigurations.

Reducing the extensive need for experts in programming robots has been a focus in research for decades. This has resulted in advances within motion, grasp and task planning, dexterous hands, etc. These advances combined into a generic action can yield highly automatic plans from only little or no expert guidance. Thereby in general reducing the need to involve experts for many specific programming tasks.

Unfortunately generic actions tend to be slow due to missing information such as where in which area objects are placed. Normally this kind of information should be specified by an expert user. However if the robot could slowly generate this knowledge itself while executing slow but robust generic actions and incrementally select more optimal and specific actions, then the system could avoid experts advice. We argue that for repetitive actions in industry this is possible.

In the remainder of this paper, we will present an approach for structuring contextual knowledge derived from previous successful action executions as depicted in Fig. 1. This knowledge is then used to choose a more specific and better parameterized action in place of the generic one. Initially we exploit that generic actions can be used to find a solution, however slow, and then gradually information gathered from the solution is used to hypothesize on the constraints of the problem.

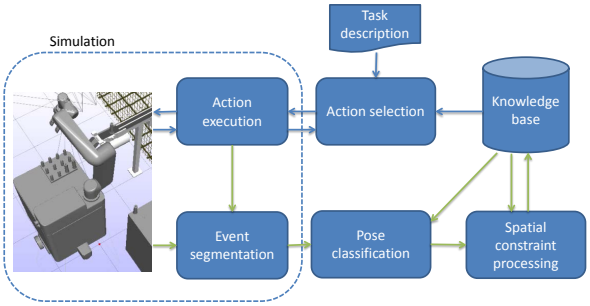

Fig. 1: Spatial constraints embedded in an action selection system. The spatial constraints are solely based on pose and object type information, feeding relevant contextual information to an action selection module that may choose actions variants based on the spatial constraints.

For representing the contextual knowledge, we propose a set of ranges or constraints over object poses in SE3. These simply define a parameterizable sub-space of SE3 which we argue can be used to optimize for better action selection or parameterization. Such a constraint could for example define the area in which we have seen specific objects being grasped or placed.

The scientific contributions of this paper include:

- presentation of a contextual knowledge representation in the form of constraints in SE3 that can be efficiently computed online and from only very limited data.
- performance comparison when using the task constraints on state-of the art pose estimation in different realistic pick-and-place scenarios.

We have limited our experiments to investigate performance on pose estimation actions. The main reason for this limitation is that we have experienced that state of the art methods for generic object and pose estimation are becoming available and that these introduce a performance bottleneck in industrial pick-and-place tasks. Furthermore, the spatial

978-1-4799-7800-7/15/$31.00 ©2015 IEEE

constraints presented in this paper, can be easily exploited to segment sensor data before doing pose estimation which drastically reduce running time of pose estimation algorithms.

After presenting the related work in Section II, we outline in Section III the individual spatial constraints and their motivation in robotics tasks. Then in Section IV we provide an overview of our combined system and describe how the constraint based pose estimation action is able to use constraints to improve its performance. In Section V we present a performance comparison on using our spatial constraints for optimizing a pose estimation action. We conclude in Section VI and discuss future directions of work.

II. Related work

This section provides an introduction to related work on structures that use constraint specifications and on work that use and represent contextual information.

Constraint analysis applied to robotics can be split in two groups: constraints on trajectory [1], [2], [3] and constraints on object pose [4], [5], [6]. Georgiev et al in [1] considered trajectory constraint detection using only proprioceptive data, joint torques of the robot's arm. In their experiments the robot exhibited exploratory behavior to learn unconstrained movement directions and favor these directions in future movements. Another example of constraint identification on motion trajectories was covered by Dragan et al. in [2]. Instead of addressing the problem of predicting a good trajectory, they addressed the problem of predicting constraints on such trajectory. Thus, specifications "above obstacle" or "left/right of object" would be translated into constraints on way-points of a linear trajectory. In [3] a method for extracting task features based on a notion of variance was proposed. Its purpose was to indicate variables that vary within a single demonstration but have little variance across multiple trials. This method allowed them to determine task constraints and also to determine a suitable frame of reference without any prior information.

Analyzing relative object orientations can reveal most constraints on object poses. Research in this area is done utilizing either task functions [4], [5] or virtual joints[6]. In iTaSC [4] the task function is defined over the pose of the object that is to be manipulated. The virtual joint method uses a 6-DOF virtual linkage spanned between the object and the robot's tool. The coordinate frames are chosen according to the physics of task that is to be performed: cylindrical for rotational symmetry, Euler angles for aligning etc. On the virtual linkage level, the constraints are represented as a triple of positions, forces and importance weights.

The use of contextual knowledge to accelerate vision tasks has been investigated in several works. In Neuropsychology it is known as contextual cueing [7], where contextual information (spatial layout and object position covariation) allows humans to perform search tasks faster. Visual cues could be absent or too numerous, so humans use visual context to find the most informative objects and regions in the scene.

A range of experiments performed by Chun and Jiang in [8] and [9] have proven that contextual cuing can be driven by spatial configuration information, objects shape and motion trajectories. In the following study Brady and Chun [10] have found that contextual cuing from local context was as strong as that observed from global context if the local context maintained its location in the overall global context. This observation corresponds to the assumption that objects may covary more frequently with items in their local context than with items in their global context. For instance, finding a sink in the scene you would limit other expected nearby objects to dishes and cutlery.

Compared to previous works this paper focuses on industrial contexts where tasks are often repeated in order to for example fill up or empty a container. The structures we have chosen to use are not novel in themselves, however combining them into and using them as contextual knowledge is novel. These structures are essentially spatial object constraints, and their motivation and derivation will be described next.

III. Spatial object constraints

This section presents a description and definition of our 4 spatial region constraints and their individual justification in the context of industrial robotics.

Our spatial regions are defined in $SE3$ and extracted based on observed (not necessarily using vision) pose information of individual objects eg. position and orientation of rigid objects in $SE3$. There are no assumptions on knowledge of the geometry of the objects, only pose information and object labels are available. The regions are defined on single object types only.

Hence, a spatial region in this work defines a limited space in $SE3$ where objects of a specific type have been detected before or where objects of a specific type are expected to occur again.

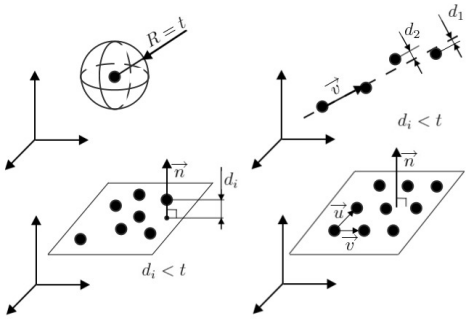

Fig. 2: The four constraint types that we use.

The object pose analysis is initialized by first computing the stable poses of the object based on the available pose information. The stable poses are essentially a classification in orientation $SO3$ and we will describe exactly what stable poses are in the context of this paper in Section III-A. We define 4 different constraint types: point, belt, table and fixture, which are depicted in Fig. 2. All can be categorized as

either structured or unstructured, where structured constraints include only objects of the same type and in the same stable pose. In the unstructured category the constraints include objects of the same type regardless of the stable pose.

We introduce stable poses of an object in the next section Section III-A and thereafter describe the four types of spatial constraints.

A. The stable pose of an Object

In the configuration space of an object, typically $SE3$, a stable pose of an object is ideally a connected subset S, where all poses in S are stable in the sense that the object will not move due to gravity, if placed in those poses. An object will thus mostly have several stable poses. Naturally, stable poses depends both on object geometry, direction of gravity and the geometry of the environment on which it is placed. We denote this supporting environment as the supporting object.

In Fig. 3 the stable poses of a pumphousing are visualized for a planar supporting object.

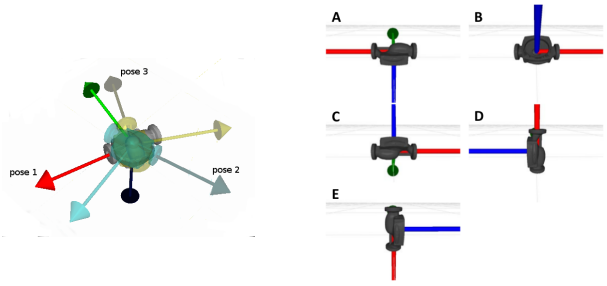

Fig. 3: To the left: Illustration of several poses of an object that belong to the same stable pose. To the right: poses sampled from 5 different stable poses of a pumphouse.

Stable poses are important spatial hints since these define a very narrow region in $SO3$ in which we expect objects to occur. Take for example the pumphouse from Fig. 3. Obviously, if we know that the supporting surface is planar then there is a very limited space in which we expect the pose to be in. Needless to say this type of information may be useful in the optimization of pose estimation methods.

In this work we use a reduced definition of stable poses focusing objects that are subjects to a planar support. In such a case stable poses can be found by calculating the invariant rotation axis of the set of object poses. This axis will always be parallel to the normal of the supporting plane. We can therefore in this work define stable poses solely based on the orientation $SO3$ of an object.

B. Point constraint

Objects placed in a fixture or in a part feeder will always have the same 6D pose or at least have the same position in space with the orientation in one of the objects stable poses. Hence, a set of poses of objects in a fixture should be constrained to a point in 6D if fully constrained or in 5D where a single degree of rotation is allowed to define the stable pose.

Hence, we define a point constraint C_{point} as a point in R^3 for a specific stable pose S_C. Any object pose p that is within a defined distance ε of C_{point} and that belong to the stable pose S_C will be part of the point constraint C_{point}.

$$C_{point} = \{p \,|\, \|\overrightarrow{p}_t - \overrightarrow{C}_t\| < \varepsilon \wedge p_R \in S_C\} \tag{1}$$

where C_t is the position of the constraint point, p_t is the position part of the pose p and p_R the orientation part.

C. Belt constraint

The belt constraint C_{belt} defines a region where object poses belonging to the same stable pose S_C occur closer than a defined distance ε from a straight line in R^3. This type of region is often observed on thin conveyor belts or for some types of part feeders which allows for easier object detection and picking.

Typically such conveyors are planar and objects on a conveyor will therefore be limited to the same stable pose with its position close to the straight line that is parallel to the conveyor belt movement.

$$C_{belt} = \{p \,|\, \|(\overrightarrow{p_t} - \overrightarrow{v}) - ((\overrightarrow{p_t} - \overrightarrow{v}) \cdot \overrightarrow{s})\overrightarrow{s}\| < \varepsilon \wedge p_R \in S_C\} \tag{2}$$

where v is a point on the line and s is the unit vector indicating the direction of the line.

D. Table constraint

Objects on wide conveyor belts or on tables are very common. The table constraint C_{table} define a region where all poses part of the constraint belong to the same stable pose S_C and are closer than a defined distance ε to a plane.

$$C_{plane} = \{p \,|\, \|(\overrightarrow{v} - \overrightarrow{p_t}) \cdot \overrightarrow{n}\| < \varepsilon \wedge p_R \in S_C\} \tag{3}$$

where $\overrightarrow{n}$ is the normal to the plane and $\overrightarrow{v}$ is a point on the plane.

E. Fixture constraint

The last constraint captures another very common structure of poses that occur in industrial contexts, namely when objects have been placed in fixtures. Fixtures often constrain objects into a geometrically ordered set of either 6D or 5D point constraints.

In this work we limit our definition of geometric order to when objects are placed in a grid in the plane. The grid define two distances and directions which define the span of the grid.

The fixture constraint impose a notion of order. This order may be helpful in deciding how to execute certain tasks like grasping or placing. Specifically what of the objects to grasp or where exactly to place the object can use the ordering derived with the fixture constraint.

$$C_{fixture} = \{p | \exists n,m \in \mathbb{Z} : \|\overrightarrow{p_t} - (\overrightarrow{v} + \overrightarrow{s_1} * n + \overrightarrow{s_2} * m)\| < \varepsilon\} \tag{4}$$

where $\overrightarrow{v}$ is position vectors of E and ε is a user defined threshold.

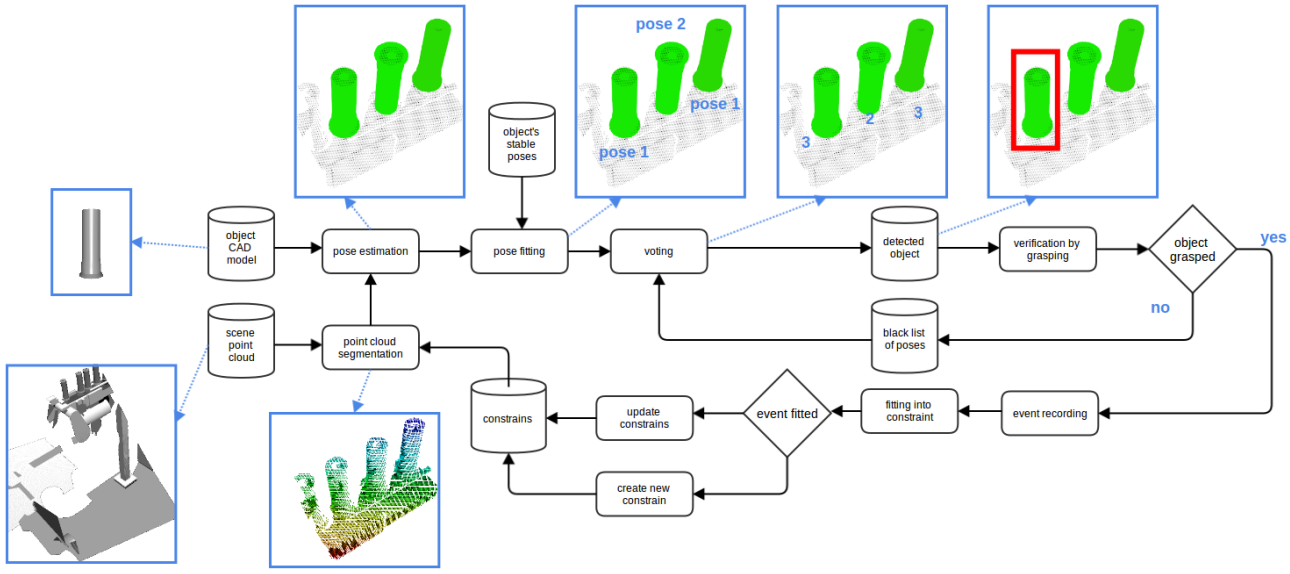

Fig. 4: An illustration of the process of determining the pose of an object. The blue boxes are the traditional pose detection that does not exploit constraints. The green boxes are the addition to the original system that enables the use of derived constraints.

IV. The Pick-and-place System

In this section, we present an overview of the system in which we perform pick-and-place operations. In the context of this paper, the hardware platform is simulated, and to keep focus on the performance gain of locating objects we have chosen to use kinematic simulation of grasping and placing actions. Hence, the focus in this section is the object pose detection action and how the constraints may be exploited by it.

In Fig. 5 we show the basic high level sequence of actions that perform a pick-and-place task. We define an action to simply be a component that defines a high level interface to robot related movements and sensing. We do not use these actions in a more elaborate framework such as presented in [11], [12]. However, it should be clear that different implementations of locate actions are interchangeable. Hence, in our experimentation we will be using two different locate actions, one that exploits constraints and one that does not.

The two locate actions are illustrated in Fig. 4. The basic flow of locating an object is depicted with the blue boxes. The green boxes indicate the addition that incorporates one way of using constraints.

The basic flow of the locate algorithm takes as inputs

- Object stable poses - This is essentially a list of the expected stable poses that objects will belong to. It is used to remove a significant amount of outliers from the general object detection and pose estimation routine.
- Object CAD - the geometric description of the object which is used for object detection and pose estimation.
- Scene point cloud - this is the point cloud image taken by the sensors of the robot system. It is in this point cloud image that objects are detected and located.

The pose estimation component does both object detection and pose estimation of detected objects based on the CAD model. In this work we consider it as a black box component which is based on the work in [13].

After the pose estimation all detected object poses are filtered according to expected stable poses (if any) and the black listing of previously tried and failed poses. The *sample-verification* corresponds to a pick attempt, however, in this work, we use the ground truth of the simulation to verify the pose of an object. False-positives are blacklisted in order to avoid repeated detections of the same false positive. All true-positive detections of objects are feed to the *event-recording* where constraint classification and creation are performed.

The *scene-point-cloud* is used directly in the pose estimation in the non-constrained based locate action. However, in the constraint based locate action the *scene-point-cloud* is first segmented using the current constraints and then

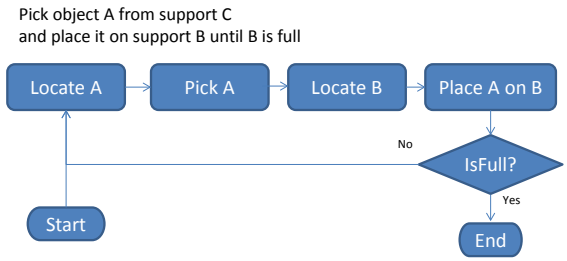

Fig. 5: The basic action sequence that we investigate. The sequence will be called continuously until some stop criteria is given.

forwarded to the *pose-estimation*. This is a simple but effective use of constraints since the size and complexity of the input point cloud negatively impact the performance of the pose estimation.

The use of the pose estimation are illustrated on two scenes in Fig. 6. The segmentation is based on a threshold ε that simply define that points further away than ε of any constraint should be removed. For a point constraint this reduce to keeping all points within a sphere with radi ε and center in the point defining the constraint. We define distance as the Euclidean distance in R^3 and not $SE3$, since objects are already ordered according to their orientation (stable poses).

For this work a fixed ε is used. The ε naturally depends on the object size which may not be known. Object property hints on size could be included using pre-defined values. Alternatively, learning of the ε could be applied by continuously changing ε and performing object detection storing successful detections and smallest ε.

V. Experimental evaluation

In this section the pick-and-place system described that was just described will be evaluated in a simulated mock-up of a typical repetitive industrial task. We will demonstrate how the fairly simple exploitation of constraints in the locate action is able to significantly speed up the task execution.

First we present the simulated setup and the task. Then in Section V-B we describe the execution system and the timing and finally we present the results in Section V-C comparing repetitive executions with and without exploitation of constraints.

A. Setup description

The experimental setup is a simulated setup and it is depicted in Fig. 7. The robot is the ”Little Helper 2” developed at Aalborg University. It includes a mobile platform Neobotix MP-L655 on which a 6-DOF manipulator (UR6-85-5-a) from Universal Robots is mounted. Finally, the gripper is a 3-finger Robotiq-3 gripper mounted on the end effector of the robot.

The mobile platform is also equipped with a RGB-D camera and a stereo camera. Both camera sensors are mounted on a 2-DOF servo platform on top of the mobile platform.

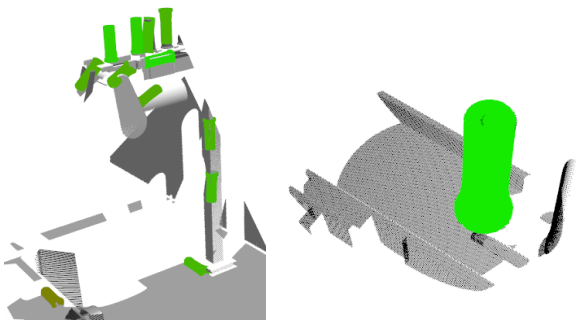

Fig. 6: Pose estimation in a full scene (left) and in a segmented scene (right). The highlighted green object CAD augmented in the detected pose.

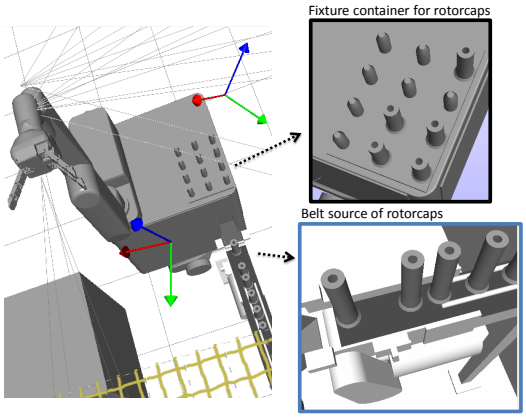

Fig. 7: To the left: a view of the robot cell. The top right is a closeup of the fixture in which the rotorcaps can be placed. In the bottom right a closeup of the conveyor belt from which rotorcaps arrive into the workspace of the robot.

The task of the robot is to move rotorcaps from the conveyor belt to the fixture. The robot can activate the refilling of the conveyor belt when it is not able to find anymore rotorcaps. The rotorcaps are located on the conveyor belt and they are all in the same stable pose but with varying distance between them.

A fixed ε for point, line, table and fixture constraint was selected to 12cm. This is approximately the same as the max distance between any two points on the rotorcap and it allowed for all points of the rotorcap to be detected in the segmented point clouds.

B. Execution system and timing

The execution system consists of executing four different actions: *locate-without-constraints*, *locate-with-constraints*, *pick-object* and *place-object*. To grasp an object, it is first located using vision. Then it is picked. To place an object, a suitable location in which to place the object is required and vision is used to locate free slots on the fixture before placing an already grasped object into the fixture.

It should be noted that these pose estimations could be optimized if we assume that only the robot is changing the scene and that only predictable changes occur. In such a case it should be sufficient to locate all (or as many as possible) objects before the first pick, remembering the position of all to avoid calling the pose estimation again. However, failed executions or the existence of human co-workers may change the scene unpredictably and therefore performing pose estimations before grasping and placing is necessary.

All actions are executed in the virtual environment. Both pick and place actions are simulated by simply moving the object for successful picks or placements. Hence, no actual motion and grasp planning is performed. Instead we assume that a successful pick takes 2 seconds which is also the case for a successful placement.

For failed pick or place action an additional penalty was added, since systems typically requires additional work in

order to recover from a failed pick. This could be due to unexpected collisions or objects that were toppled over. A failed pick was classified when an object was located wrongly by the vision system and the additional cost of a failed pick was set to 6 seconds.

The vision system grabs RGB-D images in the virtual scene and use one of the two locate actions to perform pose estimation and detection of objects. Objects are normally detected within seconds but the time depends on the size of the point cloud and the complexity of the scene. Pose estimation in a reduced point cloud as to the right in Fig. 6, is typically 5 times faster than pose estimation on a complete scene as to the left in Fig. 6.

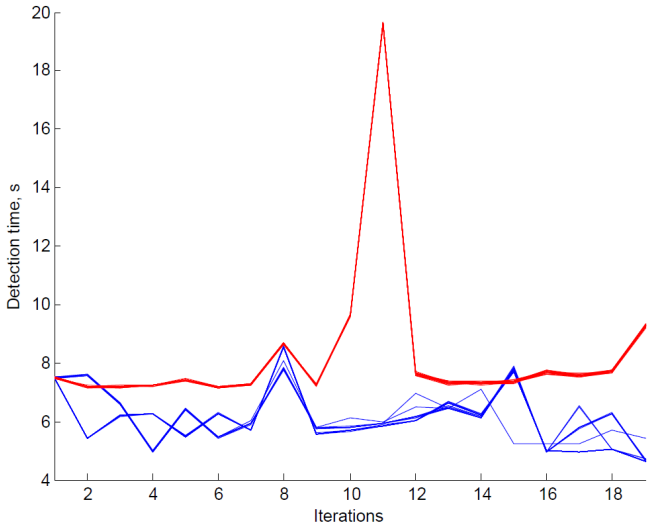

Fig. 8: Performance of pick-and-place system both with and without the use of constraints. The red indicate pick-and-place actions with generic locate action. The blue indicate pick-and-place actions that exploit constraints.

C. Results

The pick-and-place process described in the previous section V-B was used to test the performance of the constraint based locate action. The task was to fill up the fixture with rotorcaps from the conveyor belt. Initially 8 rotorcaps where placed randomly on the conveyor belt. A refilling of the conveyor belt is performed when no more rotorcaps are detectable by the system. The refiling introduce eight new rotorcaps randomly placed on the conveyor. There are no simulated time penalties added due to the re-filling.

The task of filling up the fixture from the same initial starting point was repeated 10 times for both locate actions without constraints and locate with constraints. The performance is depicted in Fig. 8 where each iteration (x-axis) is a pick-and-place attempt suggesting either a successful or failed pick-and-place operation. The time each pick-and-place operation took are plotted in the y-axis.

Both graphs start in the same point, meaning in the first pick-and-place action both methods are equally fast. This is not surprising since in the first iteration no constraints have been derived yet and the constraint based method will then default to use the pose estimation on the complete scene. After the first pick-and-place action the constraint based locate is already faster because it applies a point constraint and successfully locates a rotorcap in the segmented point cloud.

In spite of fluctuations in the time estimates it is obvious that the constraint based detection is roughly 2 seconds or 25% faster than the non-constraint based detection. The peak at the 11'th iteration is due to a large number of false-positive detections of free spots on the fixture. The object detection confuses rotorcaps with empty place holders when the fixture is nearly filled with rotorcaps. This eventually gives a large number of false positives that impact performance negatively.

The accumulated time of the locate actions are presented in Fig. 9. The non-constraint based locate is depicted to the left and it spends time in two categories:

- scene success - time spend in successfully locating objects.
- scene failures - time spend in faulty pose estimation of objects.

Evidently a majority of the time is spend on successfully locating objects. The constraint based locate action adds two additional categories.

- constraint success - time spend in successfully locating objects when using constraints for segmentation.
- constraint failures - time spend in faulty pose estimation of objects using constraints for segmentation.

Time spend in these two categories are depicted to the left in the figure. The constraint based locate action will spend time in all four categories. It does so since the non-constraint based pose estimation is used as a fall back in case no objects are detected when using constraints. The figure was compiled over 19 pick-and-place executions and it suggests that the constraint based locate action is almost twice as fast as the non-constraint based locate. There are a larger amount of failed pose estimations for the constraint pose estimation, however, the speedup in successful pose estimations easily compensates.

VI. Conclusion

In this work we presented a set of simple constraints defining regions in $SE3$. We argued that based on labeled object poses alone, we were able to quickly and efficiently derive these constraints. Furthermore we demonstrated that the constraints could be actively used to optimize a pose estimation action by using the constraints for directly segmenting the observed sensor data.

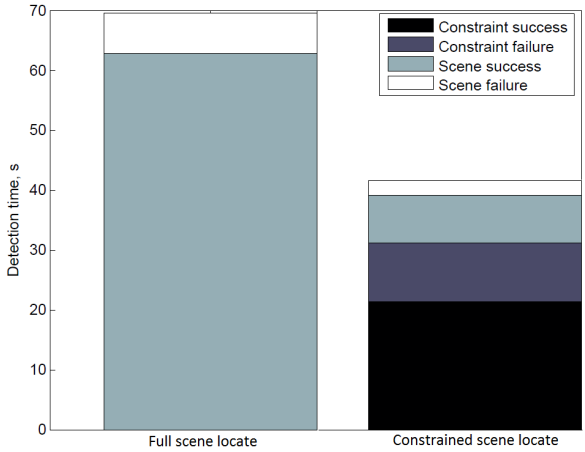

Fig. 9: The accumulated time over 19 executions of locate actions. The left bar show accumulated time of the non-constraint based locate action. The right bar show accumulated time of the constraint based locate action.

We demonstrated that the constraints significantly increases execution performance after only few pick-and-place operations and that the overall performance improvement when using rough time estimates on pick and place actions was close to 150% eg. the area between the red and blue curve in Fig. 8. Furthermore, a performance gain of roughly 200% was observed of the constraint based locate over the non-constraint based locate. Notably, the performance was only investigated in a single scenario. This benchmark should be expanded including a broader set of pick-and-place scenarios in the future.

Future work should also include investigating how to enrich the current representation with knowledge such as gravity, observed supporting surfaces and a measure on how cluttered the support is.

Furthermore, the constraints presented in this paper could be used to remove outliers in pose detection when specific constraints in an observed world is either expected or known. Such an approach holds promises for removing outliers directly from object detection data by using the representation to predict the most likely locations of objects in a scene.

Finally, the exploitation of constraints should be integrated into more actions such as the pick and the place actions. Allowing for even further performance improvements.

Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Programme and Theme: ICT-2011.2.1, Cognitive Systems and Robotics) under grant agreement no. 600578, ACAT.

The research has furthermore received founding from the Danish project Patient@home which is funded as a strategic platform for innovation and research by the Danish Innovation Fond.

REFERENCES

[1] Veselin Georgiev, Todd Wegter, Ramy Sweidan, Vladimir Sukhoy, and Alexander Stoytchev. Learning to detect spatial regions with constraints.

[2] Anca Dragan, Geoffrey J Gordon, and Siddhartha Srinivasa. Learning from experience in manipulation planning: Setting the right goals. 2011.

[3] Lucia Pais, Keisuke Umezawa, Yoshihiko Nakamura, and Aude Billard. Learning robot skills through motion segmentation and constraints extraction. In *HRI Workshop on Collaborative Manipulation*, 2013.

[4] Ruben Smits, Tinne De Laet, Kasper Claes, Herman Bruyninckx, and Joris De Schutter. itasc: a tool for multi-sensor integration in robot manipulation. In *Multisensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008. IEEE International Conference on*, pages 426–433. IEEE, 2008.

[5] Joris De Schutter, Tinne De Laet, Johan Rutgeerts, Wilm Decré, Ruben Smits, Erwin Aertbeliën, Kasper Claes, and Herman Bruyninckx. Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty. *The International Journal of Robotics Research*, 26(5):433–455, 2007.

[6] Ingo Kresse and Michael Beetz. Movement-aware action controlintegrating symbolic and control-theoretic action execution. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3245–3251. IEEE, 2012.

[7] Marvin M Chun. Contextual cueing of visual attention. *Trends in cognitive sciences*, 4(5):170–178, 2000.

[8] Marvin M Chun and Yuhong Jiang. Contextual cueing: Implicit learning and memory of visual context guides spatial attention. *Cognitive psychology*, 36(1):28–71, 1998.

[9] Marvin M Chun and Yuhong Jiang. Top-down attentional guidance based on implicit learning of visual covariation. *Psychological Science*, 10(4):360–365, 1999.

[10] Timothy F Brady and Marvin M Chun. Spatial constraints on learning in visual search: modeling contextual cuing. *Journal of Experimental Psychology: Human Perception and Performance*, 33(4):798, 2007.

[11] Joanna J Bryson. Action selection and individuation in agent based modelling. In *Proceedings of agent*, pages 317–330, 2003.

[12] Michael Beetz, Lorenz Mosenlechner, and Moritz Tenorth. Crama cognitive robot abstract machine for everyday manipulation in human environments. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1012–1017. IEEE, 2010.

[13] Anders Glent Buch, Dirk Kraft, Joni-Kristian Kamarainen, Henrik Gordon Petersen, and Norbert Kruger. Pose estimation using local structure-specific shape and appearance context. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2080–2087. IEEE, 2013.