Considering the Anchoring Problem in Robotic Intelligent Bin Picking

Diego ESCUDERO-RODRIGO [a,1] René ALQUEZAR [a]
[a] *Institut de Robotica i Informatica Industrial, CSIC-UPC, Spain*

Abstract. Random Bin Picking means the selection by a robot of a particular item from a container (or bin) in which there are many items randomly distributed. *Generalist robots* and the *Anchoring Problem* should be considered if we want to provide a more general solution, since users want that it works with different type of items that are not known 'a priori'. Therefore, we are working on an approach in which robot learning and human-robot interaction are used to anchor control primitives and robot skills to objects and action symbols while the robot system is running, but we are limiting the scope to the packaging domain. In this paper we explain how to use our system to do anchoring in *Robotic Bin Picking*.

Keywords. robot system, bin picking, generalist robots, anchoring problem, learning, HRI, symbol planning

1. Introduction

Robotic Bin Picking (RBP) means the selection of a particular item from a container (or bin) in which there are many items [3][8]. It requires a vision system (e.g. 3D laser, structured light scanner) and a robot, and they are operated in conjunction by a computer that processes the sensed data and controls the movements of the robot. RBP can be classified in: structured, semi-structured, and random; each one presents an increasing level of application complexity. In the third one items are in totally random positions, including different orientations, overlapping, and even entangled. RBP is split in three parts: item pose estimation, trajectory generation of end effector, and grasping the item.

Although more capable than ever, RBP still has its limitations. One of them is adaptability and another one is how an operator could interact with the robot system. Lack of adaptability could be seen in two different cases: (1) when we want the robot system works with new types of items, and (2) when the robot system has to deal with an unexpected situation (e.g. it is not able to pick a known item).

To provide a more general solution in which items are not known 'a priori', neither how to manipulate them, and that could deal with unexpected situations, the robot system needs to understand current situation (e.g. item overlapping) and to decide what action to execute (e.g. shuffle, ask for help). Features that we consider necessary to provide this Robotic Intelligent Bin Picking (RIBP) solution are: (a) domain knowledge based on application type, (b) robot tasks execution based on symbolic planners, (c) learning new

[1] e-mail: descudero@iri.upc.edu

concepts by interaction, and (d) improve robot interaction with operator. That is adding symbols, a knowledge base and a symbol planner.

Although classical robots, referred to as the *algorithmic approach* [1], are used in most RBP solutions, we propose to use generalist robots. A *generalist robot* acts goal-oriented by making plans, actively explores its environment, and identifies opportunities for actions. One of its capabilities is the modeling of its behavior in a symbolic representation and reasoning system. Another important point for a general solution is easy interaction between robot and operator. The robot system should be able to learn new concepts while working by itself or by interacting with the operator.

A generalist robot has a subsystem that works with symbols, and it includes a planner and a knowledge base. These symbols, objects and actions, need to be coupled to information received from sensori-motor space of the robot. And this sensor(actuator)-to-symbol problem is named as the *Anchoring Problem* [2][9]. In RIBP we have to anchor symbols that represent products (or items) in the container, predicates that represent item features (e.g. color), and actions with one or more control primitives and robot skills (e.g. close-gripper). For establishing this connection, learning techniques combined with human interaction are necessary, and it has to be done while system is running.

2. Our approach to Robotic Bin Picking in Packaging

Our objective is to develop a RIBP general solution but for the packaging domain. As we are focused on RBP for packaging, we need to characterize objects like boxes, bottles, cans, etc. and actions as pick, drop, shuffle etc. to manipulate these types of objects. To characterize objects we extract classical features such as color and geometrical properties, while to characterize trajectories that represent actions we use Dynamical Movement Primitives (DMPs) encoding positions of the joints [6].

Our system (see Figure 1) to work in the Anchoring Problem, named as ZAPS, is based on ROSPlan [7]. ROSPlan [4] is responsible of the high-level representation, and we plan to use Conceptual Spaces [5] among other techniques to connect symbolic and sub-symbolic subsystems depending on what have to be anchored. Due to its dependency on ROSPlan, the system needs that all symbols are added to the Knowledge Base (KB) before generating a plan; in case a symbol needed by the Planning System (PS) is missing the plan could not be generated. However, a symbol stored in the KB does not mean that it is anchored. In fact, it does not exist in Scene Database (SDB) in which all sensorimotor information is stored. The relationship that anchors a symbol to some data is defined inside the Anchoring Module (AM) and it is considered during plan execution.

ZAPS could be adapted to different applications by writing a specific domain file, and by selecting different set of robot skills and control primitives. In Table 1 a domain file is defined, that is types of instances that robot system knows, predicates, and what actions the system can execute depending on current state of the world.

The functionality of the system could be described in terms of action sequences that could be executed. When the PS dispatches an action, the AM receives a trigger which activates a specific anchor function. This function depends on the action dispatched and it executes one or more operations (skills or control primitives) following a predefined sequence, see Table 2 for *find-object* action.

When the action is run during plan execution, two different situations could occur: (1) action succeeds and reaches its postcondition, or (2) an operation used by the action

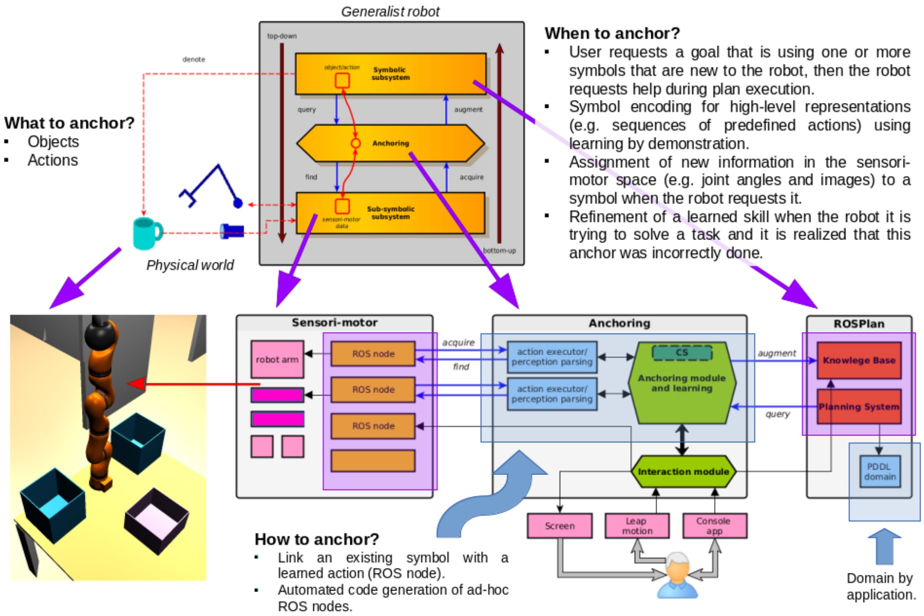

Figure 1. ZAPS: Our system to deal with anchoring in generalist robots

```
(define (domain bin_picking)
    (:requirements ... )

    (:types
        place
        object
        container - place
        box - place
        item - object
    )
    (:predicates
        (robot-at ?p - place)
        (object-at ?o - object ?p - place)
        (localised ?o - object)
        (grasped ?o - object)
        (free-tool)
        (black ?v - item)
    )
    ....
    ...

    (:action move_to_place
        ...
    )
    (:action find_object
        ...
    )
    (:action drop_object
        ...
    )
    (:action pickup_object
        ...
    )
)
```

Table 1. PDDL domain file for the experiment

fails, so some correction is needed to try to reach the action postcondition. And in any of the two situations it could happen either (a) any possible object anchored by the action fulfills the plan assumptions on the object, or (b) there is an object anchored by the action that does not fulfill the plan assumptions.

The AM manages these four cases by applying the following mechanisms: (i) anchor a new object by updating the SDB, (ii) update KB and SDB of an existing object, (iii) execute an alternative flow of operations when a correction is needed, and (iv) modify the default flow of operations used by an action

First two mechanisms are used to manage symbols related to objects, while the other two are to manage actions.

Table 2. Operations used by find-object action

Skill	Primitive
search-object	identify-object
move-camera	get-object-properties
detect-object-color	move-arm-to
shuffle	

2.1. *Managing symbols related to objects*

An object is already defined as a symbol in the KB when the system starts to execute a solved plan. However, there is not an anchor in the AM for this symbol. Therefore, when we use an operation to obtain symbol information (e.g. search-object) inside an action, the anchor is created in the AM and the SDB is updated with the information provided by this operation. If the object anchored by the action fulfills the plan assumption, next planned action is dispatched after completing the current one.

In case of an operation (e.g. get-object-properties) does not return the expected property (e.g. color) of the object, the anchor function updates the KB and SDB with this new information. Since the object anchored by the action does not fulfill the plan assumptions, the PS is triggered to cancel its current plan, solves it again with the new data and dispatches actions that solve the new plan. This behavior is a feature of ROSPlan: when current plan preconditions change, a new plan is generated and executed automatically.

2.2. *Using alternative flows for actions*

An action could fail while it is executing an operation due to different reasons, and these failures could be solved using different approaches depending on action that is being executed, see Table 3 for *find-object* action. These approaches are based on executing a different set of skills, or asking help to the operator in case the system is not able to recover by itself.

Table 3. Failures and recovery approach of find-object action

Failure reason	Recovery approach
- No object is returned by the camera	- Move camera to get other view - Shuffle objects in the container - Ask operation some help

When an operation fails during an action execution, the third mechanism is used. The AM tries to reach the action postcondition by executing a different flow of operations. So, other flows of operations must exist inside the anchor function, which are described as alternative and exception flows.

The default flow of operations describes a single path through the action, and it contains the most common sequence of operations. An alternative flow describes a scenario other than default one in which operations executed make that action succeed and reach its postcondition; it is considered an optional flow, which implies that the AM has chosen an alternative sequence of operations. Finally, an exception flow represents an undesirable sequence of operations done when the action does not succeed. An alternative flow is randomly selected by the AM. If this flow fails too, another is selected and executed. This is repeated until one succeeds or only an exception flow is available.

In case an alternative flow makes that failed action is correctly finished, the AM has two options. First one is to consider this solution as a specific solution to this specific situation. Second option is to select this alternative flow as new default flow of the anchor function. The AM could take this option if the failure situation happens several times and it is always solved by the same alternative flow. Second option is actually a re-anchoring of the action, default behavior of the anchor function is being modified. This is done by changing the code executed inside the anchor function, what is not happening when first option is selected. When the AM selects this option, it is using the fourth available mechanism we defined previously.

3. Conclusions and Future work

In this paper we have discussed what is needed to create an intelligent solution for Robotic Bin Picking. And it is clear that the Anchoring Problem must be taken into account. Our proposed system, named as ZAPS, is based on ROSPlan, which is responsible of the symbolic representation. We briefly presented how our system could be adapted to be applied to Robotic Bin Picking. Four mechanisms have been proposed to deal with problems related to the anchoring of objects and actions. Some of them involve a task re-planning while the system is running. These mechanisms are currently being tested using a simulated experiment in which items must be grouped by color in two boxes.

Our future work is to complete the experiments in different scenarios, to add logical reasoning to manipulate the knowledge base, to integrate Conceptual Spaces in the AM, and to use a real robot to allow kinesthetic learning by demonstration. In addition, we will evaluate if using a probabilistic planner in ROSPlan is interesting for our research.

Acknowledgements: This work was partially funded by Spanish research project DPI2016-78957-R.

References

[1] U. Kurup and C. Lebiere, *What can cognitive architectures do for robotics?*, Biologically Inspired Cognitive Architectures, vol. 2, 2012.

[2] S. Coradeschi, A. Loutfi, and B. Wrede, *A Short Review of Symbol Grounding in Robotic and Intelligent Systems*, Künstliche Intelligenz, vol. 27-2, Springer, 2013.

[3] D. Buchholz, *Bin-Picking: New Approaches for a Classical Problem*, Springer, 2015

[4] M. Cashmore, M. Fox, D. Long, D. Magazzeni, B. Ridder, A. Carrera, N. Palomeras, N. HurtÃşs and M. Carreras, *ROSPlan: Planning in the robot operating system*, 25th International Conference on Automated Planning and Scheduling, 2015.

[5] F. Zenker and P. Gärdenfors *Applications of Conceptual Spaces: The Case for Geometric Knowledge Representation*, Synthese Library, vol. 359, Springer, 2015.

[6] D. Escudero-Rodrigo and R. Alquézar, *Distance-based Kernels for Dynamical Movement Primitives*, Artificial Intelligence Research and Development, vol. 277, IOS Press, 2015.

[7] D. Escudero-Rodrigo and R. Alquézar, *Study of the Anchoring Problem in Generalist Robots Based on ROSPlan*, Artificial Intelligence Research and Development, vol. 288, IOS Press, 2016.

[8] C. Eppner, S. Höfer, R. Jonschkowski, R. Martí-Martín, A. Sieverling, V. Wall, and O. Brock *Lessons from the Amazon Picking Challenge: Four Aspects of Building Robotic Systems*, Proceedings of the 26th International Joint Conference on Artificial Intelligence, 2017.

[9] F. Ingrand and Malik Ghallab *Deliberation for autonomous robots: A survey*, Artificial Intelligence, vol. 247, June 2017.