

Team CVAP's Picking System at the Amazon Picking Challenge 2015

Kaiyu Hang Francisco E. Viña B. Michele Colledanchise
Karl Pauwels Alessandro Pieropan Danica Kragic

Abstract—In this paper we will present our framework used in the Amazon Picking Challenge in 2015 and some of the lessons learned that may prove useful to researchers and future teams participating in the competition. The competition proved to be a very useful occasion to integrate the work of various researchers at the Robotics, Perception and Learning laboratory of KTH, measure the performance of our robotics system and define the future direction of our research.

I. INTRODUCTION

There are three main criteria engineers evaluate when determining the need of robots in certain applications: dirty, dull and dangerous. Those are known as the 3D of Robotics. The application proposed by the Amazon Picking Challenge meets certainly the second criteria as picking and placing objects in boxes is a very repetitive and dull task with tremendous potential for automation. However, despite the defined and controlled environment the application of robots is still very challenging due to the nature of the objects to handle. In this work we present the framework we developed at the Robotic, Perception and Learning lab (RPL) in 2015 with the purpose of sharing the lessons learned with the community. First we will describe the platform used in the competition in Sec.II to motivate the strategy we adopted in Sec.III. Then we will describe the core of our system that controls the whole pipeline of actions using Behavior Trees (BTs) in Sec.IV. Then we will describe our perception module starting with the localization of the shelf in Sec.V and detections of the objects VI. Finally we will describe our grasping strategy in Sec. VII and draw some conclusion about the limitations of our system in Sec.VIII.

II. PLATFORM

We used the PR2 research robot from Willow Garage shown in Fig. ?? for the 2015 APC competition. The robot consists of two 7-DOF manipulators attached to a height-adjustable torso and equipped with parallel-jaw grippers. A pan-tilt controllable robot head is located on the upper part of the robot and equipped with a set of RGB cameras and a Kinect camera which we used in our object segmentation system. The robot is also equipped with a tilt-controllable laser scanner located right beneath the head which we also used for object segmentation purposes. A set of 4 wheels at the base provides the robot with omnidirectional mobility, which we exploited in order to control the robot back and

forth from the shelf to pick up objects and release them in the target box.

We provided some minor hardware modifications to the PR2 robot in order to address some of the challenges of the picking task, namely custom-made extension fingers for the parallel gripper in order to be able to reach further inside of the bins of the shelf and a high resolution webcam which we attached on the PR2's head in order to get a richer set of image features for our Simtrack vision system to detect the target objects in the shelf.

The robot ran on an Ubuntu 12.04 computer with a real-time linux kernel that provides 1 KHz manipulator control. All the high level task execution, perception, grasping and manipulation software components were developed under the Robot Operating System (ROS).

III. STRATEGY

In 2015 the competition consisted in picking one object from each of the 12 bins of the shelf within 20 minutes. Each bin could contain from one up to four objects making the recognition and grasping of objects increasingly difficult. In order to design our strategy we had three main limitations to consider. First, the PR2 could not reach the highest level of the shelf where three of the bins were located. Second, two of the objects of the competition were bigger than the maximum aperture of the PR2 gripper. Third, it takes the PR2 about 30 seconds to raise the torso from the lowest position to the highest. Given the time requirements that operation resulted very expensive, therefore, our strategy consisted in starting from the lowest level of the shelf giving priority to the bins with one or two objects thus minimizing the number of times the PR2 would raise/lower its torso throughout the task. Once the bins were cleared the PR2 would raise the torso to address the next level of the shelf. The more complex bins with multiple objects were left at the end disregarding the level at which they were located. Given the limitation of our robot, the known structure of shelf and the object to grasp our framework would decide what grasp strategy to apply from a pre-defined set of object-specific grasps. Once the target object was grasped, the robot would retreat from the shelf using the mobile base and release the object at a predetermined position in a box.

IV. BEHAVIOR TREES

BTs are a graphical mathematical model for reactive fault tolerant action executions. They were introduced in the video-game industry to control non-player characters, and they are now an established tool appearing in textbooks [1],

All team members contributed equally in this work.

K. Hang, F. E. Viña B., M. Colledanchise, K. Pauwels, A. Pieropan and D. Kragic are with the Robotics, Perception and Learning Lab (RPL), CAS, CSC at KTH Royal Institute of Technology, Stockholm, Sweden.

[2] and generic game-coding software such as Pygame1, Craft AI 2, and the Unreal Engine3. In robotics, BTs are appreciated for being highly modular, flexible and reusable, and have also been shown to generalize other successful control architectures such as the Subsumption architecture, Decision Trees [3] and the Teleo-reactive Paradigm [4]. In our framework, the use of BTs allowed us to have a control architecture that is:

- **Reactive:** The Reactive execution allowed us to have a system that rapidly reacts to unexpected changes. For example, if an object slips out of the robot gripper, the robot will automatically stop and pick it up again without the need to re-plan or change the BT; or if the position of a robot is lost, the robot will re-execute the routine of the object detection.
- **Modular:** The Modular design allowed us to subdivide the behavior into smaller modules, that were independently created and then used in different stages of the project. This design allowed our heterogeneity developers' expertise, letting developers implementing their code in their preferred programming language and style.
- **Fault Tolerant:** The fault tolerant allowed us to handle actions failure by composing different actions meant for the same goals in a fallback. (e.g. different types of grasps).

V. SHELF LOCALIZATION AND BASE MOVEMENT

As described in Sec. II, we used a PR2 as our robot platform. Since the arms' reachability of PR2 is relatively small in comparison to the shelf size, it is not feasible to define a fixed location for the robot to achieve the required task. As such, we have to exploit the mobility to enlarge the working space, so that the robot would be able to reach and grasp from most of the shelf bins, as well as loading the grasped objects into the order bin.

For this, shelf localization, which serves as the only landmark in the workcell, is essential for our system to guide the robot navigating between different grasping positions. Since the robot movement accumulates localization errors, it is necessary to localize the shelf in real time to close the control loop for base movement.

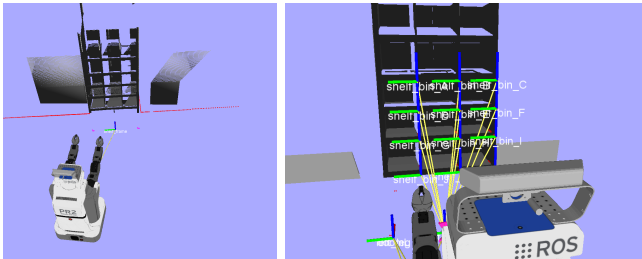


Fig. 1. *Left:* An example of shelf localization shown in rviz. The x and y coordinates of `shelf_frame` is defined as the center of two front legs, while the height of it is the same as the `base_laser_link`. *Right:* The bin frames are located at the right bottom corner of each bin.

As shown in Fig. 1, we use the base laser scanner to

localize the two front legs of the shelf. Observing that the shelf is located in front of the robot, where no any other objects are close by. Therefore, it is reasonable to find the closest point cluster and consider it as one of the legs, while the remaining cluster is considered as another leg. However, this is not a reliable procedure as there could be noise or other unexpected objects, e.g., human legs. As such, our shelf localization consists of two procedures as follows:

- **Detection:** Once the front legs are detected, before the system starts to autonomously work on assigned tasks, a human supervisor needs to confirm to the robot that the detection is correct. In case when the detection is incorrect, we need to clear the occlusions in front of the shelf until a confirmation is made.
- **Tracking:** While the robot is moving, given that we know the motion model of the robot, we update the shelf localization using a Kalman filter.

Having localized the shelf, we further estimate the shelf bin frames based on the known mesh model, see Fig. 1. As will be described in Sec. 1, knowing the bin frames will facilitate the grasp planning in our system.

VI. VISION

A. Simtrack

We created high-quality 3D textured models from all objects provided for the challenge using the approach described in [5]. To summarize briefly, each object is placed on a textured background consisting of a collection of newspapers and approximately 40 pictures are taken from various angles. The object is then constructed using Autodesk123D catch services [6] and postprocessed to separate it from the background, remove holes, reduce the mesh resolution, etc. For the texture-based detection discussed next, we also extracted SIFT-keypoints [7] from images synthesized by rendering the model from multiple viewpoints. The final object models were used for detection and grasp planning.

A large proportion of the challenge objects contained sufficient texture for keypoint-based pose detection. When such objects were present in the bin, we relied on a texture-based pose estimation method. We used high resolution images obtained from a Logitech C920 webcam mounted on the robot head. We processed a central 1152×768 pixel region of the full 2304×1536 input image for object detection. The robot head was always oriented towards the bin under consideration so that this region of interest was guaranteed to contain the objects. Our object pose detection framework, SimTrack, first extracts SIFT-features from the image and matches these to the database of SIFT features extracted from the object models in the modeling stage. In the detection stage, the objects are identified, and a rough initial pose is obtained. This pose is then iteratively refined by rendering the textured object models at the current pose estimate, and locally matching their appearance to the observed image [8]. SimTrack uses a phase-based optical flow algorithm in this refinement (or tracking) stage to reduce sensitivity to intensity differences between model and observed image.

SimTrack exploits the rich appearance and shape information of the models and correctly handles occlusions within and between objects. SimTrack is publicly available as a ROS module [9].

B. Volumetric Reasoning

We designed a fallback strategy based on 3D point cloud volumetric reasoning in case our system could not recognize the objects using SimTrack. Knowing the location of the shelf and the bins, as shown in Fig. 1, we use the PR2 tilting laser scanner mounted on the torso to build an accurate point cloud within the target bin. We use the 3D model of the shelf and its estimated location to remove any point belonging to the shelf from the constructed point cloud to obtain a cloud of the inside of a bin. Then, we apply euclidean clustering to generate plausible object hypothesis. Given that the number of objects in the bin is known, we apply our clustering strategy iteratively, increasing the distance threshold, in order to obtain as many clusters as the expected number of objects. Once the clusters are found we formulate the problem of finding the right object to pick as a bi-partite graph matching problem where the nodes on one side are the found clusters characterized by their volumes and the nodes on the other side are the 3D models of the expected objects and their corresponding volumes.

VII. MANIPULATION AND GRASPING

For robot arm motion planning we used the MoveIt! software framework and its default randomized motion planners. Even though MoveIt! provides seamless integration with ROS-enabled robots, we faced some difficulties when applying it in our PR2 APC setup. The randomized planners (e.g. RRT) often had difficulties in finding collision free trajectories to move the robot arm in constrained spaces such as the interior of the bins. Even when it did find suitable trajectories, they would often generate very large joint displacements even for small end-effector cartesian motions, which increased the chance of generating collisions with the shelf. We partially alleviated these problems by exploiting the simpler 2D kinematic motion control of the robot base e.g. to retreat from the shelf. Optimization based motion planners and a more specialized kinematic structure of the robot are possible directions of work that would address these issues.

Once the target object is detected, the object frame is obtained from our vision system. Given that the PR2 is equipped with parallel grippers, we must then decide from which position and in which direction the gripper is going to approach the object in order to then grasp the object in a secure way. For this, we always project the object frame based on the bin frame such that the X axis of the object frame points inwards the shelf, and the Z axis points downwards. Note that in order to ensure that the object can be approached without generating collisions, we manually labeled all objects offline so that the X axis returned from the vision system can always be approached with the gripper, i.e. that the gripper opening is wide enough to grasp the

corresponding side of the object. Thereafter, similarly to the grasping stages in [10], our grasping system works in the following steps:

- **Pre-grasping:** The gripper is posed in front of the bin and aligned in X axis of object frame, the gripper's opening direction is aligned in the XY plane of object frame.
- **Approaching:** The gripper approaches the object in X direction until a pre-determined depth in the object frame is reached.
- **Grasping and lifting:** The gripper closes and lifts the object to a pre-determined height. At this point, we read the gripper's joint angle to check whether the object is grasped. If not, the system will return failure and give control back to the behavior tree. Otherwise, it returns success and continue to the next step.
- **Post-grasping:** When the object is grasped, it is not trivial to plan a motion to move the arm out of the shelf, especially for large objects. Therefore, we first move the object to the bin center, and then move the whole robot backwards. Once the object is out of the shelf, the robot go to the order bin and put the object into it.

In cases when the target object is very small, e.g., tennis ball and duck toy etc., the approaching procedure shown above is not able to reach the object due to the collision with the bin bottom. In those cases, our system will first try to approach the above area of the object, and then move downwards to reach the desired grasping pose. It is worthwhile noticing that, dividing the grasping motion into steps significantly increased the success rate for MoveIt! to find solutions within a short time limitation, since the above steps efficiently guides the motion planner through narrow passages posed by the small bin areas.

VIII. CONCLUSION

We presented the framework used in the APC 2015 and the challenges that we faced. Overall, perception was the most challenging component for the competition although the challenges inherent to manipulation and grasping should not be underestimated. We can propose the following suggestions to researchers and engineers designing bin-picking robotic systems given our experience at the APC 2015:

- **Robot kinematic design.** In order to facilitate motion control of the robot manipulator and minimize the risk of generating collisions it is best to design its kinematic structure to be completely tailored to the task instead of a general purpose humanoid kinematic configuration such as our PR2 robot.
- **Grasping via suction devices.** Using suction devices for grasping would have greatly increased our grasp success rate. Suction is the de facto solution to robot grasping in many industrial applications, including robot bin picking. Part of the reason for this is that suction is in general more tolerant to errors in perception and motion control since it is less demanding in terms of

where to position the gripper relative to the object to be grasped. An object whose size is comparable to the maximum opening of a robot's parallel gripper requires very precise positioning of the gripper, otherwise the robot will unintentionally push the object away when approaching it. On other hand, suction devices have a higher likelihood of yielding a successful grasp under positioning errors as long as it is still in contact with part of the object's surface. This also means that the requirements for the perception system could be lowered, e.g. by segmenting identifying flat object surfaces instead of performing full 6D pose estimation and requiring less accurate object pose estimation.

- **Perception.** ?

ACKNOWLEDGMENTS

This work has been supported by the Swedish Research Council (VR), the Swedish Foundation for Strategic Research (SSF), and the EU FP7 program. The authors gratefully acknowledge the support.

REFERENCES

- [1] I. Millington and J. Funge, *Artificial intelligence for games*. CRC Press, 2009.
- [2] S. Rabin, *Game AI Pro*. CRC Press, 2014, ch. 6. The Behavior Tree Starter Kit.
- [3] M. Colledanchise and P. Ögren, "How Behavior Trees Modularize Hybrid Control Systems and Generalize Sequential Behavior Compositions, the Subsumption Architecture, and Decision Trees," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 372–389, April 2017.
- [4] M. Colledanchise and P. Ögren, "How Behavior Trees Generalize the Teleo-Reactive Paradigm and And-Or-Trees," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2016.
- [5] F. T. Pokorny, Y. Bekiroglu, K. Pauwels, J. Butepage, C. Scherer, and D. Kragic, "A database for reproducible manipulation research: CapriDB — Capture, Print, Innovate," *Data in Brief*, vol. 11, pp. 491–498, 2017.
- [6] "Autodesk," <http://www.123dapp.com>, accessed: 2017-05-01.
- [7] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, no. 2, pp. 91–110, 2004.
- [8] K. Pauwels and D. Kragic, "SimTrack: A simulation-based framework for scalable real-time object pose detection and tracking," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Hamburg, Germany, 2015.
- [9] "SimTrack," <https://github.com/karlpauwels/simtrack>, accessed: 2017-05-01.
- [10] K. Hang, M. Li, J. A. Stork, Y. Bekiroglu, F. T. Pokorny, A. Billard, and D. Kragic, "Hierarchical fingertip space: A unified framework for grasp planning and in-hand grasp adaptation," *IEEE Transactions on Robotics*, vol. 32, no. 4, pp. 960–972, Aug 2016.