The Importance of Being Bryce

Hang Kayu Francisco Vina Michele Colendachise 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 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 evaluates 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 could be a very repetitive and boring job. 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 develop 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 behavioural trees 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 limitation of our system in Sec.VIII.

II. PLATFORM

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 1 up to 4 objects making the recognition and grasping of objects increasingly difficult. In order to design our strategy we had 3 main limitations to consider. First the PR2 could not reach the highest level of the shelf where 3 bin 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. 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 they were located.

IV. BEHAVIOR TREES TREES

Behavior Trees (BTs) are a graphical mathematical model for reactive fault tolerant action executions. They originated in the video-game to control non-player characters, and is now an established tool appearing in textbooks [**?**], [**?**] 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 [**?**] and the Teleo-reactive Paradigm [**?**].

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 with unexpected changes in the sense that 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

VI. VISION

A. Simtrack

B. Volumetric Reasoning

VII. GRASPING

VIII. CONCLUSION

We presented the framework used in the competition in 2015 and all the challenges we had to face. With hindsight we would have brought our own robot to the competition since many of the issue we had to address were related to the robot

provided at the venue, leaving us no time to calibrate and tune our framework. Moreover the team lacked in mechanical expertise restricting our choice of the platform to the PR2, the most suited robot we had in our laboratory at the time. I would have been nice to design a specialized robot for the competition as other teams did.