System Architecture for Robotic Arm Adjustment to Different Objects in Pick-and-Place Tasks

Anatolijs Zencovs*, Arturs Simkuns, Aly Oraby, Diana Duplevska, Maksims Ivanovs and Roberts Kadikis
Institute of Electronics and Computer Science (EDI) Riga, LV-1006, Latvia
*Contact: anatolijs.zencovs@edi.lv

***Abstract*—This paper presents the architecture of the robotic system that will be used for the optimization of the production line, where a robotic arm will pick empty bottles from the box and place them on to a moving conveyor belt. One of the major contribution of this research is to simplify the addition of a new product (a bottle) to the production line by scanning the new object and passing the scan to the system. The proposed system handles all the other necessary steps for adjusting the robot (generating dataset, training, etc.), reducing the need for a machine learning expert for every product change at the factory.**

***Keywords*—Reinforcement Learning, robotic system architecture, Sim2Real, ROS2, Moveit2, pick-and-place, GAN's, Ignition Gazebo, Stable Baselines3**

I. Introduction

The system architecture provides the functional overview of each block and it's contribution to achieve defined requirements and overcome known challenges. We also utilize Reinforcement Learning (RL) [1] methods and use Generative Adversarial Networks (GAN) [2] to achieve project goals. RL has been widely used in robotics for developing autonomous systems that can learn and adapt to different environments. In the context of robot arm grasp tasks, RL algorithms together with GAN have shown promising results [3] [4] in improving the performance of robotic systems.

This work is being implemented under the IMOCO4.E project[1].

II. Problem Statement

Since this work will be implemented and tested in a factory setting, there is a number of requirements that are defined by the vendor, such as:

- All the modules including learning based modules and optionally Real2Sim perception modules need to respect real-time constraints and be deterministic.
- The whole system should be easily operated by technicians with minimal technical background and training.
- The ability to pick and place bottles with a rate of 30-45 bottles/min.
- The robot footprint should be minimal.
- The perception system should be easily adapted to new products.
- Bottles are placed randomly in a box before picking.

[1]This work has received funding from Latvian state budget under agreement No ES RTD/2021/13, H2020 ECSEL Joint Undertaking project IMOCO4.E under grant agreement No 101007311.

III. Architecture

In this section, we provide an overview of the system architecture and workflow adopted for the presented framework. This includes a description of the tools used for further training procedures and experimental setup. "Fig. 1" presents a functional overview of the system architecture used for the digital twin robot arm.

A. Dataset Generator and Perception

This block is used to generate the training dataset, detect objects from the real robot camera data and data generated in simulator, and to generate grasp poses for robot arm to pick up the detected object.

First of all, we start by generating the training dataset. We scan the bottles and generate synthetic data using the method described in [5]. The use of synthetic data makes the process of training faster and less expensive, but it also have its disadvantages. When the trained network is transferred to the physical environment and is evaluated on the real-world data, it shows worse performance than it did on the artificial data. To overcome this issue and improve the results of so called Sim2Real transfer, we use the method of processing synthetic data described in [6]. This method uses generated synthetic data together with CycleGAN [7] to make this data look more photorealistic.

B. Reinforcement Learning

RL environment is created using Gym-Ignition [8] Python package, which enables the creation of robotic environments compatible with OpenAI Gym [9]. The OpenAI Gym wrapper is used to create a gym environment for the RL agent. This wrapper enables the use of Stable Baselines3 and PyTorch for training the RL agent policy. The agent receives observations of the robot's state and environment from the gym environment and takes actions based on a learned policy. The RL agent then executes the motion plan in the Ignition Gazebo simulation using the JointTrajectoryController. The above process is repeated iteratively until the RL agent learns a successful policy for the given task.

C. Planning and control

The action planned by an RL agent is received by the Gym-Ignition package, which sends it as a desired goal to the MoveIt2 framework. MoveIt2 then uses its motion planning algorithms to generate a collision-free path for the robot to