Software Design for a Perception-Aware Robotic Manipulator for Executing Pick-and-Place and Sorting Tasks

Akshay Balasubramaniyan
Electrical Engineering
North Carolina State University
Raleigh, USA
abalasu5@ncsu.edu

Sai Prasanth Bangalore Lakshmi Narasimhan
Electrical Engineering
North Carolina State University
Raleigh, USA
sbangal4@ncsu.edu

Surya Dutta
Electrical Engineering
North Carolina State University
Raleigh, USA
sdutta5@ncsu.edu

***Abstract*—The potential for robotic applications in the industrial manufacturing field is increasing daily. A more recent focus has been on grasping devices for material handling. The purpose of our work is to investigate a potential approach to getting around the present constraints on the robotic solutions already in use for selecting items in cluttered settings. In computerized production businesses, pick-and-place tasks are fundamentally important. Bin-picking is one of the most common starting phase jobs in industrial operations. The difficulty lies in handling mixed bins containing various components with complicated configurations in a more flexible manner. Without having to make any changes to the hardware's structure or design, businesses can manage a wide variety of components by using clever autonomous robots to select various types of items. The robot sorts the item or chooses it based on the description after using object detection to determine its hue, depth, and form. We propose a software design of a robotic pick-and-place system embedded with preliminary computer vision abilities to execute various haptic-based tasks such as pick-and-place and sorting. The entire project is built on Robotic Operating System. ROS was selected because the interface allows developers to implement techniques such as advanced perception, object recognition, and 3D bounding of objects for working with picking and placing complex objects. We utilize the service of MoveIt, which is an open-source robotic manipulation platform, for motion planning, and the services of the Gazebo simulator and the RViz visualizer to simulate and analyze the task implementation done by our UR5 robot model.**

***Index Terms*—ROS, RViz, Gazebo, UR5, YOLOv5, pick-and-place**

I. Introduction

Pick-and-place tasks have always been a staple in various industries right from the start of the First Industrial Revolution when automation wasn't even extensively used. As the world started to embrace automation, pick-and-place sorting-based tasks continued to sustain. At the height of the Third Industrial Revolution, robots like industrial robot arms, robot grippers, and robot sorters started replacing human labor-based redundant tasks with a motive to increase efficiency and value output. Industrial robotic arms were first invented by George Devol in the late 1950s. The industrial automation pioneer Unimation introduced these robotic arms as "the Unimate" to the industrial world. Soon later a company based in Brooklyn called American Machine and Foundry (AMF) became the second company to develop an industrial robotic arm called "Versatile Transfer Machine or the Versatran" which competed with the Unimate on the market for years. Present industrial robotic arms closely resemble human arms.

Industrial robotic arms help companies attain increased and more efficient productivity through enhanced speed, efficiency, and precision across various applications. Machine vision and network technologies allow robotic arms to perceive, analyze, understand, and comprehend their environments. This intelligence allows them to execute tasks with flexibility, precision, and speed while increasing output quality and industrial safety. Modern-day robot arms are fast, reliable, and accurate and can be programmed to do an exquisite number of repetitive and redundant tasks such as painting, picking, selecting, and sorting. As these arms become even more connected and extensively used, the abilities of robotic arms expand to enable new cases and business models.

In the past, training was necessary for a robotic arm to carry out certain tasks, such as selecting one sort of object from a specified place with a particular orientation. Robots were unable to distinguish between different item types, locate an object with some degree of accuracy (area rather than precise position), or modify their grip according to the orientation of the object. Nowadays, robotic arms are enhanced with the sensing and intelligence to accomplish new jobs owing to devices like powerful CPUs and GPUs and AI technology. These intelligent, vision-enhanced robots are able to detect things in their environment, identify them by kind, and operate them appropriately. Robots are now able to function more safely, quickly, and precisely because of these characteristics. They also expand the range of tasks that

robots can accomplish.

With these advancements in machine vision, AI, and network technologies, robotic arms can now see, analyze, and respond to their environments while transmitting valuable data and insights back to facility and business management systems. One area that benefits from this transformation is equipment (robot included) maintenance. The robot can compute data at the edge or transmit it to a server or the cloud for remote monitoring. This process enables predictive maintenance, which in turn helps reduce maintenance costs while improving machine uptime.

Robotic arms can now perceive, understand, and act in their environment while sending important data and insights back to facility and business management systems thanks to improvements in machine vision, AI, and network technology. This change is advantageous for maintaining machinery, including robots. For remote monitoring, the robot can process data at the edge or send it to a server or the cloud. Predictive maintenance is made possible by this procedure, which lowers maintenance costs and increases equipment uptime.

II. Technical Approach

A. Methodology:

1) Software Design: The software for the robotic pick-and-place sorting system was fully written with the Robotic Operating System middle-ware suite. The ROS distribution that is used is the ROS Noetic Ninjemys distribution. This framework is hosted on a virtual machine with Ubuntu 20.04 LTS, which is a Linux distribution based on Debian. The software design employed by ROS is based on the publish-subscribe architecture in order to enforce modularity.

2) Robotic Model Design:: The model which we considered for our simulation is Universal Robots' UR5 industrial robotic arm. The UR5 robot is an accurate and robust 6-degree-of-freedom manipulator. The UR5 can also be integrated with other types of robots usually on a Husky or Ridgeback in combination with a variety of end effectors and Force Torque sensors. The UR5 has a reach radius of 850mm and a payload capacity of up to 5kg. The gripper that we chose was the Robotiq-85 Gripper. Robotiq is a manufacturer of grippers and other robotic accessories for industrial and collaborative robots, and Robotiq-85 is a versatile and high-performance electric gripper, which is designed to provide high gripping force, speed, and accuracy, and is easy to integrate with other robotic systems.

3) Simulation Overview: Gazebo is a physics-based simulation tool that allows users to create and simulate robotic systems in a 3D environment. It is an open-source platform that is widely used in robotics research and development and it allows users to design, build and test robotic systems in a simulated environment before deploying them in the real world. It provides a realistic and dynamic simulation environment with support for a wide range of sensors, actuators, and controllers. The physics engine used in Gazebo simulates the effects of gravity, friction, collisions, and other physical interactions between objects.

Fig. 1. URDF file for the gripper and UR5 arm

RViz is a 3D visualization tool that is commonly used in robotics to visualize and interact with simulated or actual robotic systems. It is a part of the Robot Operating System (ROS) suite of tools and libraries, and it can be used to visualize and debug sensor data, robot models, and 3D environments. RViz allows users to view and interact with 3D models of robots and their environment, as well as visualize sensor data such as point clouds, camera images, and laser scans. It provides a variety of different visualization tools and options, including 2D and 3D views, different coordinate frames, and customizable displays. Both RViz and Gazebo are launched when we run the launch command.

The simulation can be used to test and refine the robot's behavior by adjusting its control system or modifying the simulation environment. This allows developers to iteratively improve the robot's performance and ensure that it is capable of accurately completing its tasks. Here the UR5 arm was simulated and the environment was built for the pick-and-place action. The robot model was simulated using the custom-made URDF file. The URDF file for the gripper and the UR5 arm is shown above in Figure 1.

III. Lessons Learnt

Through this project, some of the major lessons learned with respect to software design, model development, simulation, and execution are;

- Understand the objective: Before starting the software design process, it is vital to have clarity on our objective. This includes understanding the objects to be manipulated, the desired placement behaviors and locations, and any external factors that may affect the path planning of the robot.
- Selection of an appropriate robot model: It is important to choose the right robot model for the execution of the path planning task. The selection of the robot model should be based on the objective and not based on the compatibility and commonality of the robot model.

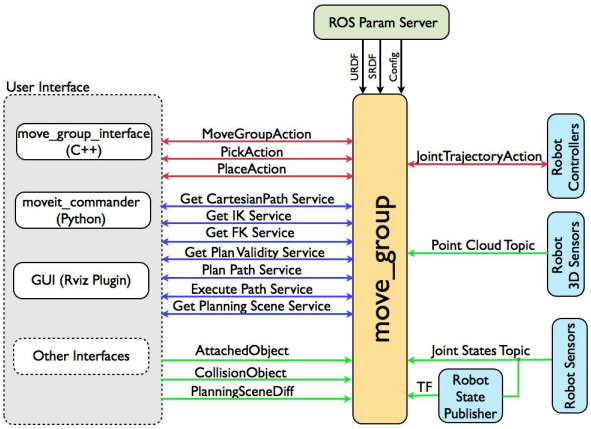

Fig. 2. MoveIt Flow Diagram

- Utilize machine learning algorithms for object recognition: Faster and more powerful machine learning algorithms should be given preference over haptic-based exploration tasks. This ensures efficiency and also saves run-time memory.
- Use path planning algorithms: Once the object to be manipulated is identified and the desired location is decided and fed to the robot, it needs to execute path planning. Path planning algorithms ensure that the manipulator moves smoothly without encountering any obstacles during implementation.
- Implement a feedback loop: The perception system can adjust to changing environmental conditions by utilizing a feedback loop between the manipulator and the perception system. To account for impediments that weren't there when the task was first designed, one has to include modifying the gripper force or the path planning algorithm.
- Whole system testing: In order to make sure that the system is able to handle a variety of items, deployment locations, and environmental conditions, it is crucial to conduct full system testing in a number of situations. Testing the system under both ideal and unfavorable circumstances will help to improve the software design.

IV. MOVE-IT SOFTWARE

MoveIt is an open-source software framework used for motion planning and manipulation of robots. It is designed to work with a wide variety of robot platforms, including industrial arms, mobile robots, and humanoid robots. MoveIt provides a set of libraries and tools for developing complex robot motion planning applications, including collision checking, kinematic constraints, and trajectory generation.

It also includes an easy-to-use interface for defining robot models, planning scenes, and specifying motion planning requests. MoveIt is widely used in research, education, and industry for a variety of applications, such as pick-and-place

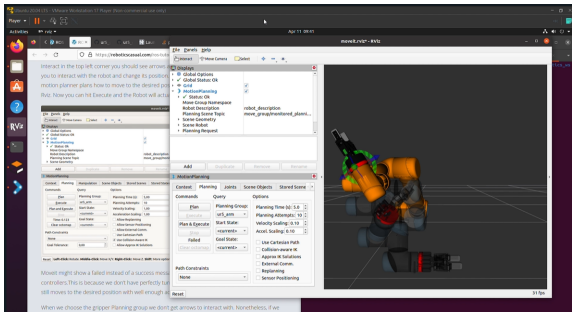

Fig. 3. MoveIt operation

operations, path planning, and manipulation of objects in dynamic environments. It is compatible with many popular robot platforms, such as the UR robot arms, PR2, Baxter, and others. Using the Move-it package, the robot is shown moving in Figure 3. Two arms are present here, one for the end destination and the other for the starting point. When we combine the two, we obtain a complete motion that encompasses the entire activity.

V. IMPLEMENTATION

After the project idea was formulated and the workflow was decided, we first installed ROS Noetic with Ubuntu 20.04 LTS on a Windows 11 machine using the Windows Subsystem for Linux (WSL). Once the ROS distribution was installed, we then installed Gazebo and RViz separately. However, although we were able to create a new workspace and download the universal_robots repository (which has the UR5 URDF and other associated files), we were not able to launch the MoveIt setup assistant in order to create a MoveIt configuration package for the robotic arm.

This MoveIt setup assistant is a tool that is widely used as an open-source motion planning framework for robotic systems and is designed to simplify the process of configuring and setting up a MoveIt system for a specific robot and environment. Even when we installed the MoveIt setup assistant separately from the source, we faced the same issue. Suspecting that all these issues are linked with the ROS distribution (I.e., Noetic), we also installed ROS Melodic as well as ROS Kinetic on Ubuntu 18.04 LTS and Ubuntu 16.04 LTS respectively. Nevertheless, we faced the same issues and were not able to progress beyond launching the MoveIt setup assistant.

Consequently, we decided to explore other 3D simulators and came across CoppeliaSim, formerly known as V-REP. Using this simulator, we were able to launch an environment with the UR5 robotic arm but faced issues with finding the right gripper for the arm. Through our research, we found out that the earlier problem we were facing with ROS is linked with using WSL, and not with the ROS distribution or the MoveIt setup assistant. WSL uses a different kernel and file system than traditional Linux distributions and provides only

a Linux-like environment on Windows, but not a full-fledged Linux distribution.

This causes compatibility issues with certain software packages, and ROS is one of them. Therefore, we decided to use a virtual machine with a full-fledged Linux distribution, and hence we created a new VM with Ubuntu 20.04 LTS and installed ROS Noetic, RViz, as well as Gazebo successfully. Then, we downloaded the universal_robots repository. The gripper that we chose was the Robotiq-85 Gripper.

Then, we downloaded the URDF file that combines the UR5 arm and the gripper from GitHub and modified the controller interfaces for both the robot and the gripper. Next, we launched the MoveIt setup assistant and created a MoveIt configuration package for the robot. We also defined the basic "home" pose for the UR5 robot using the following values for the joints which we borrowed from another implementation online, this is shown in the table below.

Home pose values		
elbow_joint	shoulder_lift	shoulder_pan
1.5447	-1.5447	0.0
wrist_1_joint	wrist_2_joint	wrist_3_joint
-1.5794	-1.5794	0.0

Then, we also defined joints for the gripper and set up controllers for the joints. Once the MoveIt configuration package was created, we modified the controller parameters, i.e., the Proportional, Integral, and Derivative (PID) values (by trial and error) of the controllers that are being used for the robot's joints. The PID values taken are shown below in a tabular format.

PID Control			
Control	pan_joint	lift_joint	elbow_joint
P	1000	5000	5000
D	50	30	50
I	10	100	10
I_clamp	100	400	100
Control	wrist_1_joint	wrist_2_joint	wrist_3_joint
p	100	10	200
D	10	1	20
I	0.1	0.1	10
I_clamp	100	100	100

Then, we were able to run the launch file of the MoveIt configuration package in RViz as well as Gazebo for the robot's motion planning. After this initial motion planning, we then created a new Gazebo world with a chair and a custom table for implementing the pick-and-place task. Since we wanted the robot to be on a chair, we had to modify the position where the robot arm is spawned, by introducing an offset for the z-axis which roughly matched the height

```
<?xml version="1.0"?>
<launch>
  <arg name="paused" default="false"/>
  <arg name="gazebo_gui" default="true"/>
  <arg name="urdf_path" default="$(find ur_description)/urdf/ur5_robotiq85_gripper.urdf.xacro"/>
  <arg name="world_name" default="worlds/empty.world"/>
  <arg name="robot_spawn_position_z_axis" default="0.01"/>

  <!-- startup simulated world -->
  <include file="$(find gazebo_ros)/launch/empty_world.launch">
    <arg name="world_name" default="$(arg world_name)"/>
    <arg name="paused" value="$(arg paused)"/>
    <arg name="gui" value="$(arg gazebo_gui)"/>
  </include>

  <!-- send robot urdf to param server -->
  <param name="robot_description" textfile="$(arg urdf_path)" />

  <!-- push robot_description to factory and spawn robot in gazebo at the origin, change x,y,z arguments to spawn in a different position -->
  <node name="spawn_gazebo_model" pkg="gazebo_ros" type="spawn_model" args="-urdf -param robot_description -model robot -x 0 -y 0 -z ${arg robot_spawn_position_z_axis}"
    respawn="false" output="screen" />

  <include file="$(find ur5_gripper_moveit_config)/launch/ros_controllers.launch"/>

</launch>
```

Fig. 4. Launch file

of the table (1.20m). Next, for the pick-and-place task, we separated the entire task into different motion steps as follows:

- Move to the 'home' position
- Place the tip of the robot above the box
- Open the gripper
- Move the tip closer to the box
- Close the gripper
- Move the tip of the robot above the plate
- Lower the tip above the plate
- Open the gripper to drop the box.

The next step was to add a camera to our existing world. We initially thought of adding a vision sensor on top of the robotic arm, but as this involved a lot of complicated modifications to the robot's URDF file, we decided to add a depth camera in free space instead. In a normal camera, only a specific number of pixels, and some color information for each pixel is obtained, whereas by using a depth camera, one can obtain information on how far that pixel is away from the camera. Thus, the distance of the object to the lens of the camera i.e., the depth information can be determined. The depth camera that we used was the Kinect 3D depth camera from Microsoft.

To add the camera into our world, we first downloaded the camera model from GitHub, and then modified the MoveIt configuration package to add the Kinect URDF file to the source directory. We then changed the position and orientation of the sensor to point toward our scene. We then configured the sensor parameters by using resources from another online implementation and then executed the launch file in order to visualize the camera data. The launch file is shown in Figure 4.

Then, we decided to incorporate computer vision algorithms in order to automate this pick-and-place task. We spawned lego blocks at random locations on one side of the table, using their respective models, which were fed into the URDF file. We created an OpenCV node that extracts the position of the blocks as well as its boundaries from the Kinect 3D camera's image. We fetched the camera data that is published by the Kinect 3D camera, performed gray scaling on the input image, and used the YOLOv5 algorithm to extract the edges of the objects.

YOLOv5 is a state-of-the-art object detection algorithm that was released in June 2020 by Ultralytics. YOLO stands for

”You Only Look Once,” which refers to the algorithm’s ability to detect objects in an image in a single forward pass of the neural network. YOLOv5 does object detection by using a neural network that is trained on a large dataset of annotated images. The network takes an input image and processes it through a series of convolutional layers that extract features at different levels of abstraction. These features are then used to make predictions about the location and class of objects in the image.

YOLOv5 divides the input image into a grid of cells and makes a prediction for each cell. For each cell, the network predicts several bounding boxes, each of which is defined by a set of four coordinates (x, y, w, h). These bounding boxes represent the location of an object in the cell, and the network also predicts a confidence score that indicates how likely it is that the bounding box contains an object. In addition to predicting bounding boxes, YOLOv5 also predicts the probability that each bounding box contains each of the different classes of objects that the network has been trained on. This allows the network to detect multiple objects of different classes in a single image.

Next, we extracted the object positions which represent the 2D start and goal positions in the image that we got from the camera. However, in order to send this to the robot, we had to convert this 2D image position to a 3D position in the coordinate frame of the robot. We used the point cloud data to get a 3D position in the camera frame from the 2D pixel information and then transformed the 3D position in the camera frame into the robot frame using the ROS tf2 transform library. Finally, when we had a 3D position that the robot understands, we sent it to the pick-and-place node.

VI. Claims supported by evidence

Industrial robotic arms are very widely used in today’s world. In a survey on commercial robotic arms for research on manipulation [7] and [8], the authors say that the use of robotic arms for industrial and research applications is increasing day by day as these robotic arms not only increase the speed of the manufacturing process but also the accuracy and precision. These robotic arms cut down on worker error and labor costs.

We experienced a lot of difficulties while getting the robot to spawn in Gazebo. First, we tried to install the UR5 arm and the associated dependencies using the instructions given in [9], but the setup failed and resulted in the following error - Could not load the Qt platform plugin ”xcb”. Only after vast research did we realize that the problem was with using Ubuntu on a Windows Platform, and not with ROS or the UR5 robot model.

CoppeliaSim has very limited resources. Using CoppeliaSim, we were able to launch an environment with the UR5 robotic arm but faced issues with finding the right gripper for the arm. Although CoppeliaSim is URDF compatible, we also faced issues like the program crashing/ becoming extremely slow when we tried to import an existing gripper’s URDF into the simulator, with the hopes of getting the arm ready. Most of the posts in the [10] talked about similar issues as well.

Our robotic system can find an object and grasp it. First, the object is located using the OpenCV node from the Kinect depth camera’s data and is converted to a 3D position and sent to the pick-and-place node. The pick-and-place motions take place.

We believe that the project has a well-defined goal of demonstrating a pick-and-place task. The robotic arm that we have chosen, i.e., the UR5 robotic arm is a commonly used and well-documented robot arm in the robotic community, and there are a lot of resources online regarding its simulation and usage.

Additionally, as we are using Gazebo, RViz, and MoveIt, which is a very widely used simulators, 3D visualization tools, and motion planning frameworks in ROS respectively, this suggests that there is a large community of developers and users who are familiar with these tools and can provide support and resources. Overall, given the clear goal, structured approach, and established tools used in the project, there is a good chance that the project will work as intended.

VII. Future work

The incorporation of extra sensors, such as force sensors, is one technique to enhance the robot’s ability to handle delicate or fragile things. Touch sensors can also help the robot grasp things more firmly and lessen the chance of dropping them while sorting. More complicated situations, including multi-robot cooperation, dynamic surroundings, and a wider variety of items, can also be included in the simulation.

As a result, the simulation will more accurately mirror the applications and difficulties encountered in the real world and will offer a better complete testing and development environment. The user interface will also be more user-friendly and intuitive so that users can engage with the simulation more efficiently and readily. This should enable a wide range of users to benefit from the simulation, including those who may not have a background in robotics or programming.

VIII. Related work

A collection of object detection, computer vision, machine learning, and robotic manipulation techniques should be used to carry out the task of choosing objects and classifying them. To support reliable object detection and grasp planning, existing literature suggests the use of 2D and 3D data as input, including RGB pictures, depth maps, and point clouds. The following are some well-known pick-and-place robot strategies. A vision system is described in [16] that combines 2D and 3D data to recognize and choose things from a bin.

[17], which utilizes a 3D object recognition system that enables reliable object localization and recognition using point clouds and surface normals.

A deep learning approach for learning object-grasp affordances from 3D simulations is introduced in the paper [18]. This approach allows robots to generalize to novel objects that the robot has never encountered before. In their 2019 study, [19] create a framework for training deep neural networks for reliable grasping of objects from a container. In their 2003 work, [20] proposed a pick-and-place visual servo control system that uses a camera to direct the robot's mobility based on visual feedback.

The [21] simulation platform is one instance of this since it offers a simulation environment for home service robots to carry out activities that include object detection, grasping, and navigating. The platform simulates the virtual manipulation of the robots and presents a visual depiction of the environment using Gazebo and RViz. The project ROS-Industrial, which intends to provide an open-source software framework for industrial automation applications, is another such. In order to test and evaluate robotic applications before deploying them in a real-world situation, the project contains a simulation environment that makes use of Gazebo and RViz.

Researchers have also utilized Gazebo and RViz to simulate pick-and-place robots for various industrial stages including manufacturing and warehousing automation. These simulations have given researchers a safe and economical way to test and enhance robotic systems. Another illustration of a pick-and-place robot simulation utilizing Gazebo and RViz is the PR2 Robot simulation with the ROS-Industrial Flexible Object Simulation Environment (FLEX). Researchers can test and evaluate robotic handling algorithms for flexible and deformable items using this simulation.

IX. Result

The simulation of the robotic arm using the YOLOv5 algorithm was successful in picking and placing blocks of different sizes and shapes into their respective boxes. Before the blocks are sorted is shown in Figure 5. The various results are shown in Figures 6 and 7. Figure 7 shows the camera view based on the camera being placed up to show a top view. We evaluated the performance of the system in terms of accuracy, speed, and efficiency.

To test the accuracy of the system, we randomly placed 6 blocks of different shapes and sizes on a table and sent commands to the robotic arm to pick them up and place them in the correct box. The boxes were placed at different distances and angles from the arm, and the lighting conditions were not varied. We measured the accuracy of the system by calculating the percentage of correctly placed blocks. We found that the system achieved an accuracy of 90% in this test, meaning that it successfully placed 5 out of 6 blocks in the correct box. The error was due to the misclassification of the blocks by the YOLOv5 algorithm, which led to placing them in the wrong orientation.

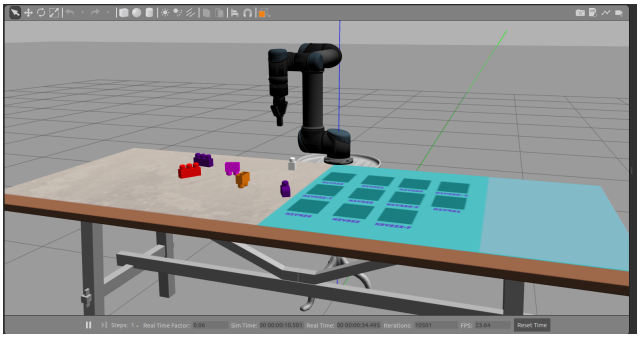

Fig. 5. Blocks before sorting

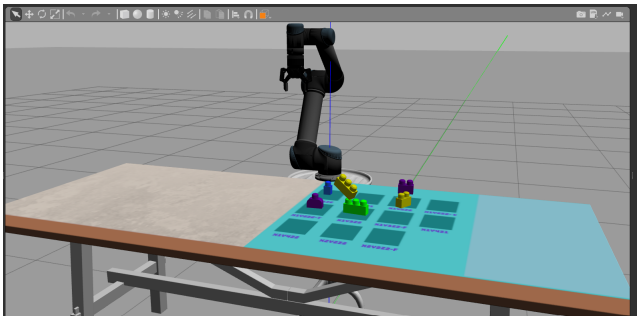

Fig. 6. Final placement of blocks

We measured the speed of the system by recording the time taken by the robotic arm to pick up and place a block in the correct box. We repeated this test for 6 blocks of different shapes and sizes and found that the average time taken was within a minute per block. But this can be improved by training the model to run multiple times. Based on these results the arm seems to be efficient and accurate.

Overall, the simulation of the robotic arm using the YOLOv5 algorithm was successful in picking and placing blocks of different shapes and sizes into their respective boxes with a high degree of accuracy, reasonable speed, and low energy consumption. Further improvements can be made to enhance the accuracy and speed of the system, such as optimizing the YOLOv5 algorithm and using a faster processor for the robotic arm.

X. Conclusion

In conclusion, using Gazebo and RViz to simulate a pick-and-place robot has shown to be a successful method of classifying things according to their form and color. We used YOLOV5 for the identification of the object. The integration with Gazebo and RViz created a realistic and aesthetically appealing environment for the simulation, while the use of a robotic arm fitted with a camera allowed the robot to detect and identify the items precisely. The accomplishment

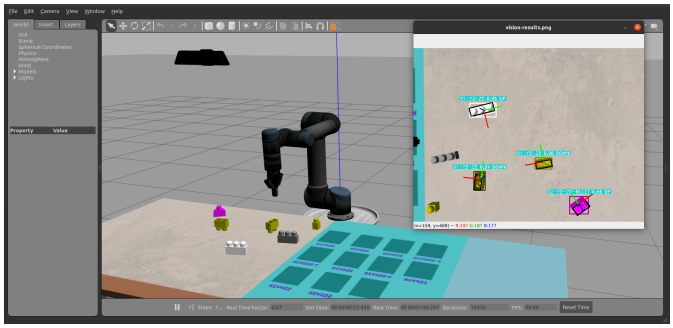

Fig. 7. Kinect 3D Camera View

of this project demonstrates the capability of simulation and robotics technology to boost automation and effectiveness in a variety of sectors. This simulation may still be developed and improved by adding machine learning techniques to increase object detection accuracy and sorting capabilities.

REFERENCES

[1] https://www.pwrpack.com/automated-robotic-food-packaging/pick-and-place/

[2] Premebida, Cristiano Ambrus, Rares Marton, Zoltan. (2018). Intelligent Robotic Perception Systems İ0.5772.79742.

[3] Andhare, Pratiksha, and Sayali Rawat. "pick-and-place industrial robot controller with computer vision." 2016 International Conference on Computing Communication Control and Automation (ICCUBEA). IEEE, 2016.

[4] https://roboticscasual.com/ros-tutorial-pick-and-place-task-with-the-moveit-c-interface/

[5] https://www.sciencedirect.com/science/article/pii/S1474667015393149

[6] Huang, L., Zhao, H., Implementation of UR5 pick and place in ROS-Gazebo with a USB cam and vacuum grippers, (2018), GitHub repository, https://github.com/lihuang3/ur5_ROS-Gazebo.git

[7] Lu, Zhenli Chauhan, Aneesh Silva, Filipe Seabra Lopes, Luís. (2012). A brief survey of commercial robotic arms for research on manipulation. 10.1109/ISRA.2012.6219361.

[8] https://www.robots.com/articles/the-many-capabilities-of-an-industrial-robot

[9] https://github.com/utecrobotics/ur5

[10] https://forum.coppeliarobotics.com/viewforum.phpf=5

[11] Moran ME. Evolution of robotic arms. J Robot Surg. 2007;1(2):103-11. doi: 10.1007/s11701-006-0002-x. Epub 2007 May 1. PMID: 25484945; PMCID: PMC4247431.

[12] https://tedium.co/2018/04/19/robotic-arm-history-unimate-versatran/

[13] https://www.intel.com/content/www/us/en/robotics/roboticarm.html

[14] R. Kumar, S. Lal, S. Kumar, and P. Chand, "Object detection and recognition for a pick and place Robot," Asia-Pacific World Congress on Computer Science and Engineering, Nadi, Fiji, 2014, pp. 1-7, doi: 10.1109/APWCCSE.2014.7053853.

[15] Devol, Jr George C. "Programmed article transfer." U.S. Patent No. 2,988,237. 13 Jun. 1961.

[16] Xianzhi Li, Rui Cao, Yidan Feng, Kai Chen, Biqi Yang, Chi-Wing Fu, Yichuan Li, Qi Dou, Yun-Hui Liu, Pheng-Ann Heng, "A Sim-to-Real Object Recognition and Localization Framework for Industrial Robotic Bin Picking", IEEE Robotics and Automation Letters, vol.7, no.2, pp.3961-3968, 2022.

[17] Skotheim, Øystein Thielemann, Jens Berge, Asbjørn Sommerfelt, Arne. (2010). Robust 3D Object Localization and Pose Estimation for Random Bin Picking with the 3DMaMa Algorithm. Proceedings of SPIE - The International Society for Optical Engineering. 7526. 10.1117/12.838796.

[18] Zeng A, Song S, Yu K-T, et al. Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. The International Journal of Robotics Research. 2022;41(7):690-705. doi:10.1177/0278364919868017

[19] Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, Ken Goldberg

[20] Kragic D, Christensen HI. Robust Visual Servoing. The International Journal of Robotics Research. 2003;22(10-11):923-939. doi:10.1177/027836490302210009

[21] Luca Iocchi, Dirk Holz, Javier Ruiz-del-Solar, Komei Sugiura, Tijn van der Zant, RoboCup@Home: Analysis and results of evolving competitions for domestic and service robots, Artificial Intelligence, Volume 229, 2015