Chista RoboCup@Work 2020 Team Description Paper

R. Ghasemi[1], S. Salehi[1], M. Abdolianpour[1], M. Asadsangabi[1], M. Javanmardi[1],

M. Zareii[1], A. Zarei[1], F. Mehri[1]

1 Robotics Research Center, Chista Academy, Shiraz, Iran.

Email: suhair_salehi@yahoo.com

Websise: http://chist.org/

Abstract. This paper presents the CHISTA@Work team. We describe the current state of robot hardware and software as well as plans and goals toward 2020 RoboCup@Work competition. The software architecture is based on ROS framework with LabVIEW. The research of team is focused on improving the current algorithms to increase robustness with LabVIEW.

Index Terms—RoboCup [at]Work, Team Description Paper, ROS framework, LabView.

1 Introduction

The CHISTA team was founded in 2009 at the Robotic Center of Chista academy by graduate and undergraduate students in fields of electronics, computer science, and mechatronics engineering. The focus of the team has been Autonomous robot so far, but we decided to concentrate on robots with industrial usage. Our short time goal before RoboCup 2019 competitions is to attend RoboCup IranOpen 2018 competition in @work league. Over the years we participated in different national and international events. An industrial robot interacts with people, complex objects, and dynamic environment. So, we are trying to improve our perception, manipulation, and fault detection algorithms to improve our robot's functionality and We are also trying to upgrade this robot by Artificial intelligence algorithms and methods.

2 Hardware

2.1 Base platform

To make navigation simpler for the robot, the base platform is designed and implemented as a "Holonomic" system that has four mecanum wheels. Mecanum wheels designed by the Rhino program

and printed with 3dPrinter with PLA materials. To control the base of our robot we use the MyRio 1900 [1], Arduino [2]Mega 2560, Arduino Nano and Raspberry pi which connects to ROS [3]and LabVIEW [4]through a serial port. This board also performs the task of battery management, odometry estimation, and emergency stop. The movement speed of the base is limited to a maximum linear velocity of 20 cm/s. These are not the absolute maximum rates available for the robot, and it can be changed via a parameter if needed. In our tests, these values were fairly sufficient. All the high-level control and vision software are running on a laptop. It lets the robot to move easily without any off-board process requirement, but to increase the safety of operation, another laptop can be connected to the main one.

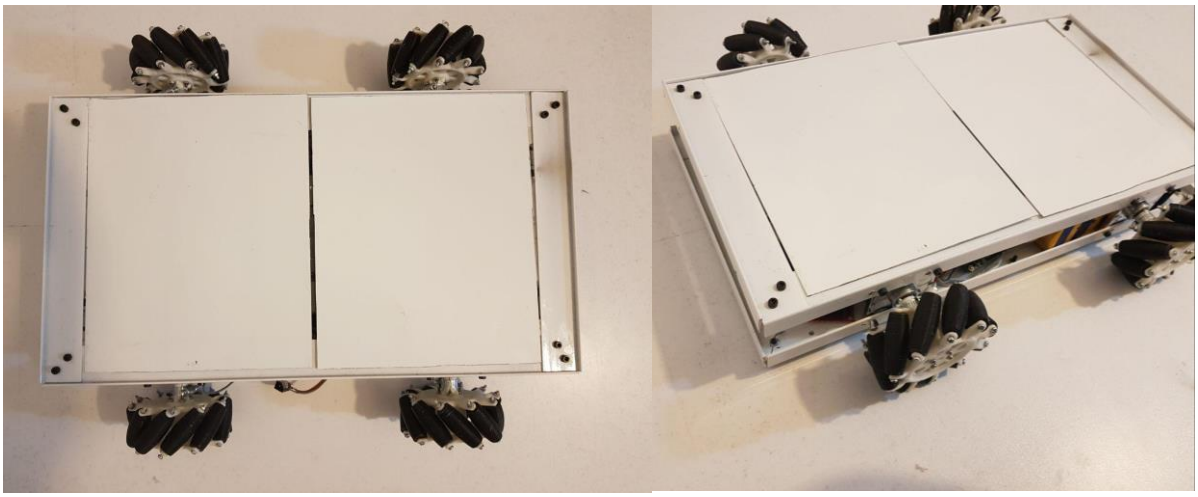

Figure 1.: Chista base platform

2.2 Manipulator and Gripper

We designed an arm based on the open source platform uArm [5]with a number of modifications. We used PLA to build it and replaced the motors with the DC Motor to increase torque and arm accuracy (see Figure 2). Finally, we redesigned the end operator so we have two other factors. The arm is designed not to endure itself, but this can be done with the software. The arm is capable of lifting more than 350 grams and has two servo motors for the gripper. The team is currently working on arm control and object detection using sensors on the gripper.

[1] To read more about MyRio 1900 please refer to this link: http://www.ni.com/en-us/shop/select/myrio-student-embedded-device?modelId=125751

[2] To read more about Arduino please refer to this link: https://www.arduino.cc/

[3] To read more about ROS please refer to this link: http://www.ros.org/

[4]To read more about LabVIEW please refer to this link: http://www.ni.com/en-us/shop/labview.html

[5] To read more about uArm please refer to this link: http://ufactory.cc/#!/en

Figure 2.: uArm manipulator (a), Our manipulator (b)

2.3 Sensors

Our robot is equipped with both Kinect and RGB-D camera, which is used in navigation, perception and recognition tasks. We use the Kinect360 (see figure 3) to implement SLAM. We also need a camera to identify objects and markers. The three distance SRF sensors are also used to provide more assurance in the front of the robot.

Figure 3: Kinect 360

3 Software

The software framework is based on ROS (Robot Operating System) toolkit on LabVIEW. ROS has a core that manages inter-process communications among ROS nodes. This will allow us to work on separate, standalone packages for different purposes and then, interconnect them via ROS. Python and LabView has been used for its programming , as well as for image processing from LabView and OpenCv, and is currently working on a deep neural networks for faster detection and classification.

3.1 Navigation

The navigation is based on ROS navigation. First, we use gmapping to create a map with the kinect and odometry data. we did use SLAM was utilized to localize the robot. The location of robot is then verified with AR marker detection algorithms. The LabVIEW toolkits was used to achieve this goal to navigate through the map, we use Move-Base that works based on two cost maps for local and global planner. To choose the path we used the A* algorithm has been implemented and interface by the LabVIEW. After the trajectory is generated, we define mid goals on it. The base platform tries to reach the final goal by passing through each mid-goal in the defined order. The use of omnidirectional movement enables the robot to move to these goals easily in three steps: 1. rotate toward the path, 2. move on the path, and 3. rotate toward goal position (see Figure 4). It also been assisted by SRF sensors to detect nearer obstcales.

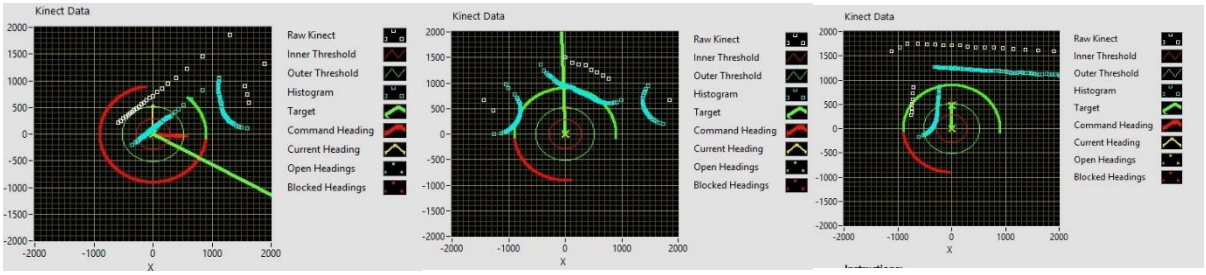

Figure 4: Simulation of our navigation software

3.2 Object perception and recognition

we use the LabVIEW toolkits to detect and recognize the objects. First, we position the camera on top of table so it gets the top-down look to objects. Then we convert the image to gray-scale and we use Morphology Finally, we use adaptive template matching to classify them. (see Figure 5) demonstrates this routine and also OpenCv and deep learning is currently being used to help identify objects.

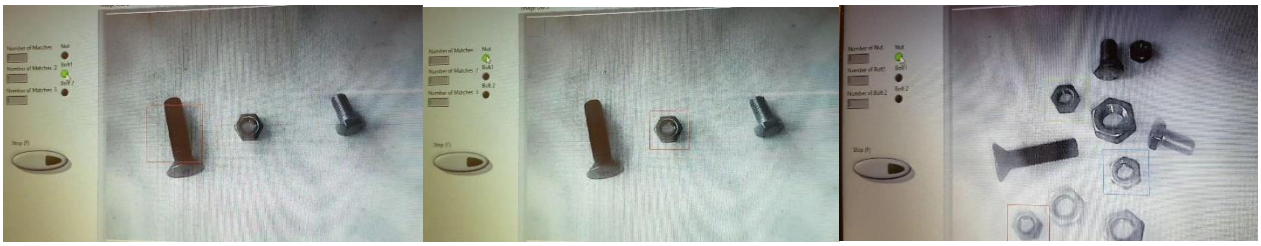

Figure 5: Object recognition

3.3 State machine

The state machine acts as an action client, which sets the goals in navigation and manipulation to accomplish the tasks and receives feedback in case of issues. Since our software architecture is based on LabView, different VI are used. receive and read sensors data and then analysis and sent to microcontrollers and microprocessors system which are MyRio and Arduino firstly camera data have been processed for adjustment and matching will be filtered and segmentaded kinect will send a data to navigator to extract and determine the useful data Then we use the LabVIEW to connect to ROS. All other VI are developed entirely by the team.

References

1. Tateno, K., Tombari, F., Laina, I., & Navab, N. (2017). Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 6243-6252).
2. Whelan, T., Kaess, M., Johannsson, H., Fallon, M., Leonard, J. J., & McDonald, J. (2015). Real-time large-scale dense RGB-D SLAM with volumetric fusion. *The International Journal of Robotics Research*, *34*(4-5), 598-626.
3. Posada-Gómez, R., Sandoval-González, O. O., Sibaja, A. M., Portillo-Rodríguez, O., & Alor-Hernández, G. (2011). Digital image processing using LabVIEW. In *Practical Applications and Solutions Using LabVIEW™ Software*. IntechOpen.
4. Relf, C. G. (2003). *Image acquisition and processing with LabVIEW*. CRC press.
5. Karan, B. (2015). Calibration of kinect-type RGB-D sensors for robotic applications. *Fme Transactions*, *43*(1), 47-54.
6. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436.
7. Viola, P., & Jones, M. (2001). Robust real-time object detection. *International journal of computer vision*, *4*(34-47), 4.