Integration of Vision based Robot Manipulation using ROS for Assistive Applications

Rajesh Kannan Megalingam
Dept. of Electronics and Communication Engineering
Amrita Vishwa Vidyapeetham, Amritapuri, India
rajeshkannan@ieee.org

Rokkam Venu Rohith Raj
Dept. of Electronics and Communication Engineering
Amrita Vishwa Vidyapeetham, Amritapuri, India
rohithraj353@gmail.com

Tammana Akhil
Dept. of Electronics and Communication Engineering
Amrita Vishwa Vidyapeetham, Amritapuri, India
akhiltammana1234@gmail.com

Akhil Masetti
Dept. of Electronics and Communication Engineering
Amrita Vishwa Vidyapeetham, Amritapuri, India
akhilmasetti03@gmail.com

Gutlapalli Nikhil Chowdary
Dept. of Electronics and Communication Engineering
Amrita Vishwa Vidyapeetham, Amritapuri, India
g.nikhilchowdaryatp@gmail.com

Vignesh S Naick
Dept. of Electronics and Communication Engineering
Amrita Vishwa Vidyapeetham, Amritapuri, India
vsnayak160@gmail.com

***Abstract*— Providing vision to robots has been an effective application in the field of robotics which enhanced automation. There have been several algorithms developed in recent times for object recognition and YOLO v3 which is one of the best has shown very good results in real-time object recognition. Consequently, the same has been chosen for this research. This research aims at developing a vision based autonomous manipulation system for assistive applications such as service robots, rescue robots, wheelchairs, etc. A detailed system integration between vision and robotic arm using the Robot Operating System (ROS) framework has been presented. This paper also explains the method of coordinate transformation between camera and manipulator reference systems and the results are discussed.**

Keywords—YOLO V3; object recognition; coordinates; transformtaion; arm motion

I. INTRODUCTION

In this technical era, automation has been the most sophisticated advancement which enhanced the research in automated robotics and its applications to much higher levels. In this aspect, Human aided robotics has been one of the most rapidly developing technologies. Some of the robust features of this domain includes object vision and robot manipulation. The vision based manipulation has solid applications in human aided robotics. The method of such integration is explained in this research. This research uses YOLO v3 as the object recognition algorithm to recognize the trained objects and determine its coordinates for the manipulation purpose.

Kinova robotic arm has been used for manipulation in this research. The different control modes of kinova such as admittance, Cartesian, velocity and force or torque control etc. has aided applications in various fields such as service robotics, physical assistance, medical applications, mobile manipulation, rehabilitation, telecommunication, etc. [1]. This paper elaborately explains the method of developing a vision based manipulation system using the ROS framework. The paper also presents a detailed flowchart and transformation of coordinates implemented in the developed system. The different parameters to be considered for object manipulation are also discussed.

II. MOTIVATION AND PROBLEM STATEMENT

Disabilities can be of different types such as vision impairment, dumb, physically handicapped, or crippled. Besides the daily increase in technology, the number of physical disabilities has also been increasing. As per the World Health Organization (WHO) report on disability and health in 2018, about a billion people i.e. nearly 15% of the world population have some form of disability [2]. According to WHO nearly 1% of the world population i.e. 65 million people approximately needs a wheelchair [3].

Physically disabled and elders face difficulties in performing their daily activities at their residences. They need personal assistance to help them in doing the regular tasks. Humans can solve this issue but labor scarcity and cost are the major problems in many countries. To address this problem, an autonomous system is developed by integrating computer vision with the modern Kinova robotic arm to implement assistive robots to help the handicapped and elders in the home and hospital environments.

III. RELATED WORKS

Research paper [4] deals with the integration of speech and image which results in an artistic technology by the combination of art and technology and this paper states some methods to implement using artificial intelligence. Paper [5] is a research work that explains the mathematics involved in controlling a 3 DOF manipulator using vision data which has a wide area of applications. Research paper [6] explains an

efficient method of tracking and analyzing the moving objects using the ROS and Simulink software. Paper [7] presents an improved accuracy of YOLO v3 for UAV detection for anti-UAV applications. The author in paper [8], used crawlers for YOLO v3 training through which images are collected for proper pre-processing. Paper [9] explains the implementation of YOLO v3 for baggage detection in public places using GUI to make the system invariant of light and camera position.

Positioning and tracking applications of a robotic arm using laser data is clearly demonstrated in paper [10]. Robotic arms can be deployed in many assistive applications, one of such typical examples is implemented in paper [11] where a robotic arm is designed and implemented for a coconut tree climbing robot. Paper [12] explains the keyboard control and RVIZ simulation of a 6 DOF robotic arm in ROS. Paper [13] came up with an idea of adapting the Euclidean cluster extraction algorithm to recognize the objects i.e. if the arm can't reach the position it will change its position and grasp the object. Implementation of vision and Arm as a part of a complete system in service robot is dealt in paper [14]. Research paper [15] explains the integration of vision and manipulation by attaching a camera to the end effector of a manipulator suitable for industrial applications. The research in the journal [16] clearly explains the application of Capsule Neural Network that divides an image into tiny segregations and preserves every minute details. Unlike CNN which cause loss of some information, capsule neural networks has shown better efficiency in different applications.

IV. System Architecture

A. *Hardware Architecture*

Fig. 1 depicts the complete infrastructure required for this research. A laptop with Intel i5 processor and NVIDIA graphics GeForce GTX 1050 installed with the ROS Kinetic framework on Ubuntu 16.04 is used as the processor for the system.

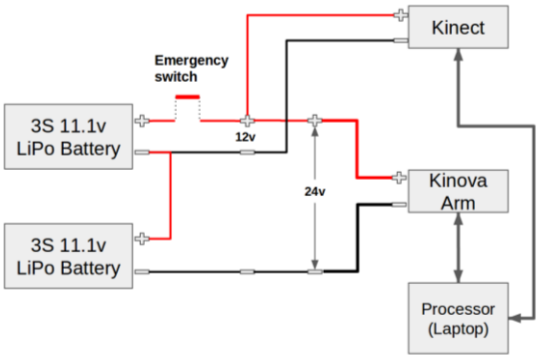

Fig. 1. Hardware Architecture

The Kinect v2 RGB depth camera with RGB Field of Vision (FoV) of 84.1° × 63.8° and depth FoV of 70.6° × 60° is used for vision. The operating range of this vision sensor is in the range of 4-4.5 meter. The 6 DoF Kinova Jaco (j2s6s300) whose reach length is 0.7m from its second DoF has been used as the manipulator arm. A pair of 3 cell LiPo batteries rated at 11.1v to a maximum of 12.4v each and 10,000 mAh capacity has been used as the power source. The maximum output voltage rating is 12.4v. The highest power consuming unit is the Kinova arm which needs 24v for its operation and hence the pair of batteries are connected in series. Kinect needs 12v for its functionality which is directly drawn from one of the batteries. An emergency switch is connected in series with the power source in order to shut down the robot in case of failure.

B. *Software Architecture*

Fig. 2 represents the software architecture of the system based on the ROS framework with two primary independent nodes. They are Kinect and Manipulator nodes which are connected to the ROS master. Each double-sided arrow indicates two different topics via which the two nodes communicate with each other via ROS master. The Kinect node publishes the required object coordinates to the ROS master which transforms these coordinates with reference to the manipulator according to which the manipulator navigates in its workspace. The detailed system integration is explained in further sections.

Fig. 2. Software Architecture

The whole system or process involves five main steps:

1. Object recognition with YOLO v3
2. Object parameter analysis
3. Assigning object Id
4. Coordinates transformation
5. Manipulator motion planning

1. Object recognition and training with YOLO v3

YOLO v3 is a convolutional neural network (CNN) for object detection and recognition in a real-time system. The algorithm applies a single neural network to a single image at once, applies filters to it and then gives the output. It takes at least 200 to 250 images to train a single object using YOLO v3. All those picture used to train a single object should differ from each other in orientation and should be of good resolution for an effective training. After collecting the dataset of images of required object, they are labelled and are trained, and using the custom weights file, the images are being recognized.

Object recognition:

The YOLO algorithm is based on CNN which consists of different layers such as convolution layer, ReLU layer (activation function), pooling layer and fully convoluted layer. An input image to be detected is passed through all these layers and the final output is obtained as a vector. The input image is passed over the convolution layer where the features of the image are extracted. Basically, the convolution layer process the image, pixel by pixel whose feature values are stored in a matrix that will be used for further computation. This resultant matrix is sent to the ReLU layer where the values below

threshold limit will be equal to zero. Here, the activation function is the ReLU layer which is,

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$

Next, the pooling layer compresses the matrix with respect to the window matrix which is assumed one. Finally, in order to get the perfectly flattened matrix these three layers run in a loop. The matrix is given through the fully convoluted layer where the image is being classified and output is obtained. The highest number in the vector of that particular position is compared to the trained vector and when the highest position of vector is matched correctly then the object is recognized.

Object coordinates:

Consider the drawing box of an object as in Fig. 3 whose 3D Coordinates has to be determined in the camera or the Kinect reference systems.

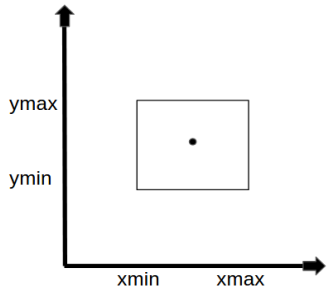

Fig. 3. Drawing box of an image

The coordinates (P_x, P_y, P_z) where, P_x and P_y indicates the 2D position of the mid-point of the object at the depth distance of P_z from the camera. These coordinates are determined from the following equations (1, 2 & 3):

$$P_x = [(X_{center} - ppx) * depth]/f_x \quad (1)$$

$$P_y = [(Y_{center} - ppy) * depth]/f_y \quad (2)$$

$$P_z = Grayvalue \quad (3)$$

Where ppx and ppy are the principal points and f_x and f_y are the focal lengths along the X and Y axes respectively which are taken in the Kinect reference system. Here, Kinect v2 camera is used whose camera intrinsics are as following:

$\{f_x, f_y, ppx, ppy\} = \{594.21, 591.04, 339.5, 242.7\}$.

Object training:

The daily life objects have been used for testing the proposed system. In order to recognize them, the system first needs to be trained with a dataset of different images of each object. The objects are trained by implementing the YOLO v3 custom object dataset. In order to train each object, different images of a particular object have to be considered which are labelled using the annotation tool. Different images based on the orientation, size, color, etc. are taken for each object. For this research, different number of images have been taken according to each class which is represented in the Table.1 where the number of images required to train a particular object depends on evenness, symmetricity and geometry of an object. A weights file is obtained which is our custom objects dataset.

Table.1. Training data for each object

Object	Number of images
Bottle	310
tomato	295
Cup	300
dining table	280

2. Object parameter analysis

The objective of the system is to meet the demands of vision-based manipulation applications with utmost accuracy. For any such application, there are certain object parameters to be considered. In a given set, the objects shall differ in their shape, size, weight, hardness, and fragility. The size and shape of an object will vary the orientation of the gripper required to pick or place it. The weight of an object should fall under the payload capacity of the manipulator. The hardness of the object determines the maximum force that a gripper can exert on it, to hold it properly. Next comes the fragility factor, though some objects are strong they are fragile enough to break if the applied force is more than the threshold force. That threshold force is the maximum force that can be applied to an object without breaking. This threshold force changes from object to object based on weight, geometry, brittleness, and smoothness of the object surface. To achieve this, the object details such as shape, size, hardness, and brittleness are also published to the ROS master along with object Id. Here in this scenario, three different objects are tested; they are bottle, cup and fruit.

3. Assigning object Id

While training the images there is a need for specifying each class name for each object i.e. the name of the object. In this scenario 4 objects have been trained: bottle, tomato, cup and dining table. These object names are taken as the serial numbers called as object_id. When the object is recognized its object_id is given correspondingly. The object_id is then published to the arm through ROS messages. In the Table. 2, different object_id is assigned to each trained object.

Table.2. Objects and assigned Id's

Object name	Assigned Id
Bottle	0
tomato	1
cup	2
dining table	3

4. Coordinates transformation

Fig. 4 represents the difference in coordinate systems of Kinect camera and the Kinova arm. Consider O_c and O_m are the origins of the Kinect sensor and Kinova manipulator respectively. Whereas C_{oc} represents the object coordinates in

O_c reference system. C_{ce} represents the position of Kinect w.r.t O_e and C_{oe} represents the destination of the end-effector w.r.tO_e.

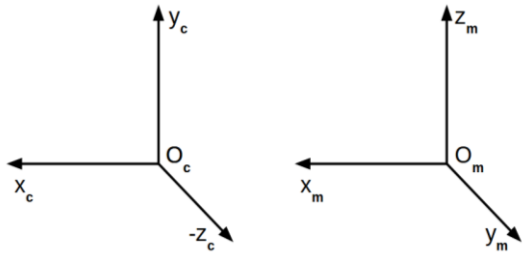

Fig. 4. Kinect and Kinova reference systems

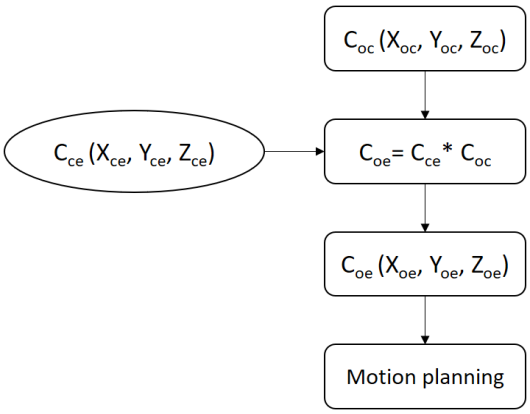

Fig. 5. Coordinates transformation

The origin of end effector i.e. O_e and the origin of the kinova manipulator O_m are not the same. There exists another transformation that is related to the kinematics of the manipulator. C_{ce} can be determined only after finding the C_{em} by using the manipulator kinematics.

The coordinates of the selected object published by the Kinect i.e. C_{oc} are with respect to the origin of Kinect reference system. The Cartesian coordinates to where the end effector has to be reached i.e. C_{oe} is with respect to the origin of the Kinova manipulator. If the origins of Kinect and Kinova are the same, then$C_{oe} = C_{oc}$. If there is a linear transformation between O_c and O_m then,$C_{oe} = T_{lin} * C_{oc}$. If there is a change in orientation between O_c and O_m then,$C_{oe} = T_{rot} * C_{oc}$. In this scenario, it is assumed that O_c and O_m differ in both linear and orientation. Therefore, the required transformation involves both linear and rotational parameters nonzero. Fig. 5 represents the flow of coordinates transformation method implemented and the equation (4) represents the mathematical representation of coordinate transformation from the Kinect reference system to the kinova reference system.

$$\begin{bmatrix} X_{oe} \\ Y_{oe} \\ Z_{oe} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & X_{ce} \\ 0 & \cos\theta & -\sin\theta & Y_{ce} \\ 0 & \sin\theta & \cos\theta & Z_{ce} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_{oc} \\ Y_{oc} \\ Z_{oc} \\ 1 \end{bmatrix} \quad (4)$$

Where the *4x4* matrix represents the transformation matrix in which the first 3 columns indicate the rotational transformation whereas the last column transforms the coordinates linearly. θ is the rotation of O_c with respect to the $\boldsymbol{x}$-axis of O_e which also means that the axes X_c and X_e are collinear. The θ also represents the tilt in Kinect orientation w.r.t $\boldsymbol{x}$-axis. Therefore, the rotational transformation is confined to YZ-plane unless there is panning in camera orientation i.e. rotation w.r.t $\boldsymbol{y}$-axis.

5. Manipulator motion planning

Kinova, being one of the high-end assistive manipulators, uses inverse kinematics for its efficient motion planning. There are available packages in GitHub to control the manipulator. This method is implemented by using the MoveIt motion planning framework which uses inverse kinematics to control the manipulator. MoveIt takes input the pose values as $(\theta, \phi, \Psi, x, y, z)$ where, (θ, ϕ, Ψ) represents the orientation values whereas (x, y, z) represents the desired position values. The motion planning is implemented in such a way that the end effector moves to the desired position as the difference between current and desired positions. MoveIt follows the plan and execute mechanism i.e. once the input is published to MoveIt, it plans the path first then executes the motion. The figure 6 shows the path planning in the ROS RViz visualization window whereas the figure 7 represents the manipulator reached its desired pose by executing its motion. The orientation (θ, ϕ, Ψ) will vary based on the image parameters such as height, width, size, and shape. The position coordinates obtained from the YOLO v3 are further transformed into the manipulator reference system for an effective motion planning.

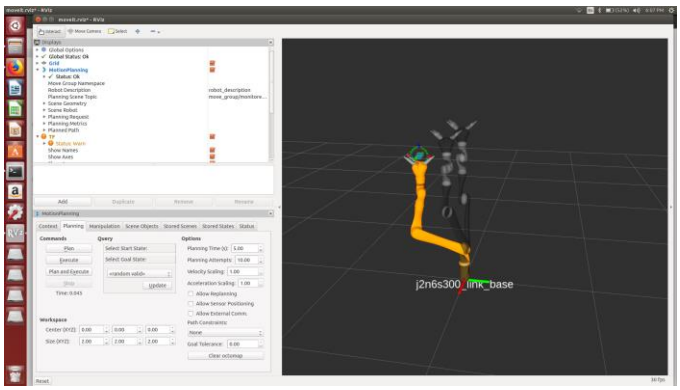

Fig. 6. Visualization of Path planning in RViz

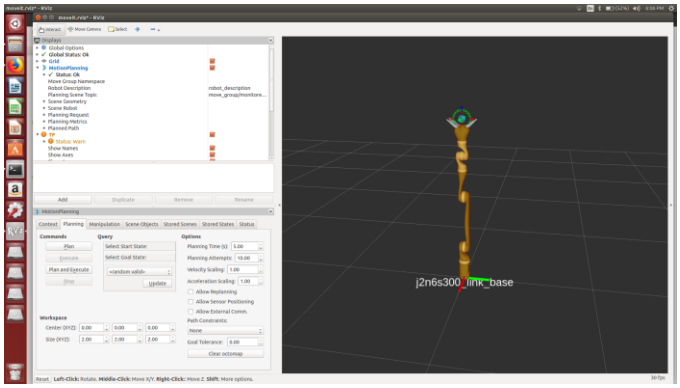

Fig. 7. Visualization of Motion execution in RViz

V. System Integration

The flow of the process is as in the flowchart represented in Fig. 8 that represents the different steps involved in the system integration. The overall system integration is done using the ROS framework, a meta-operating system that provides data transmission between the process, device control, and packet management. In this research, "pose stamped" is a type of message that has 7 floats, a string, time, and uint32 data types. Before transforming the object coordinates from the Kinect to end effector reference system, the coordinates of the end effector and camera w.r.t kinova reference system i.e. C_{em} and C_{cm} respectively need to be determined. C_{em} can be obtained from the kinova arm package by using the kinematic modeling. C_{em} has to be determined manually from the setup or the implemented system.

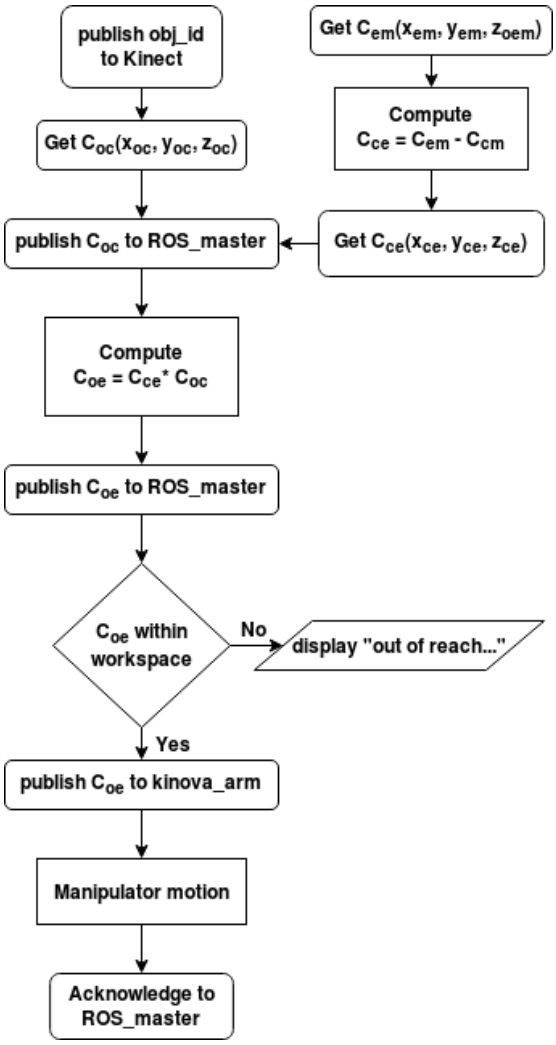

Fig. 8. Flow chart of the implementation system

The next step is to determine the camera coordinates w.r.t end effector i.e. $C_{ce} = C_{em} - C_{cm}$. Next step is the master has to subscribe the object coordinates from YOLO v3 i.e. C_{oc} based on the user input object_id. This C_{oc} is in the camera reference system and has to be transformed as C_{oe} by using the aforementioned transformation technique. This new coordinates C_{oe} are to be published from master to the kinova package. Hence the new pose values of the arm to be published to the MoveIt package are$(\theta, \phi, \Psi, X_{oe}, Y_{oe}, Z_{oe})$.

VI. Results and Discussion

The testing of this system requires a simple setup. The kinova arm is fixed at the end of a table and the Kinect has been placed such that its position w.r.t the end effector i.e. the coordinates$C_{ce} = (0.2, 0.1, -0.1)$. The objects were placed on a dining table in front of the system setup where the trained objects were randomly changed in position and sometimes replaced with other objects. When a user inputs a valid object_id, the task starts as explained in the integration section. Several experiments were conducted to evaluate the integration of the system. The test results of experiments were tabulated as in the Table. 3.

The experimental work has observed that the proposed system integration is successful in all the experiments despite the limitations: the reach length of the arm and the accuracy of YOLO v3. To check if the coordinate C_{oe} is within manipulator's workspace, the distance is given by$\sqrt{(x_{oe}^2 + y_{oe}^2 + z_{oe}^2)}$. Fig. 9 shows the recognition of trained objects used in the setup. In the Fig.10, the classification accuracy of the objects used for the experimentation are depicted. Here, the accuracy is absolute for the cup but when it comes to dining table it is 56% and for bottle and tomato the accuracy observed is 77% and 72% respectively. However, the classification accuracy increases with more number of test cases and depends on the features used in the implementation of YOLO. In addition to this, more data can also be added to train the objects to enhance the accuracy. As in the Table.1, it shows the number of images considered for the particular objects. Fig. 11 shows the obtained coordinates from YOLO v3 based on the input object_id or frame_id which is '1' and accordingly, the required pose values for manipulator has been displayed in the camera reference system.

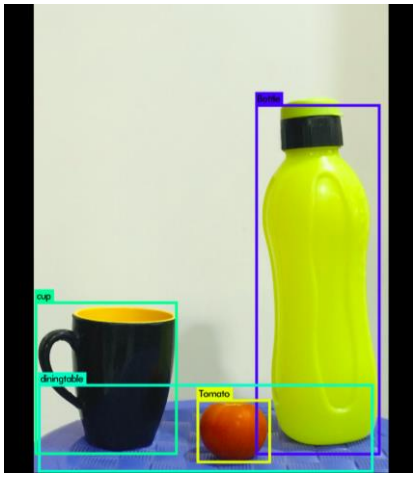

Fig. 9. Object recognition

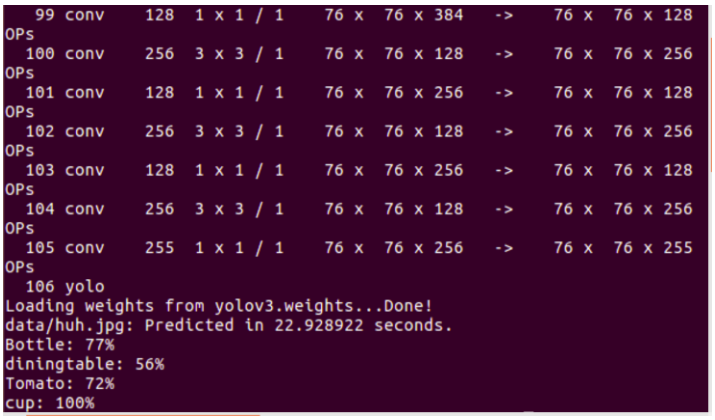

Fig10. Classification accuracy of the objects

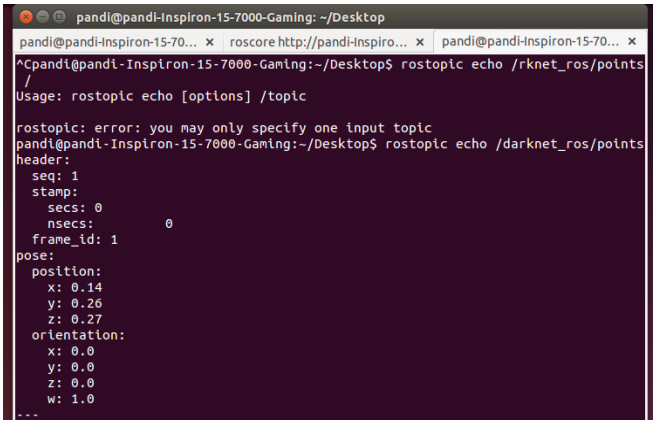

Fig11. Coordinates from YOLO v3 based on object_id

Table. 3. Test results of different objects and remarks on the object recognition and arm motion

Expt. No	Object_id	Object placed	Object recognized	C_{oc} (x_{oc}, y_{oc}, z_{oc})	C_{oe} (x_{oe}, y_{oe}, z_{oe})	Reach (m)	Object recognition?	Reached the object?
1	0	bottle	bottle	(0.24, 0.12, 0.31)	(0.44, -0.21, 0.02)	0.4879	Done	Yes
2	1	tomato	tomato	(0.11, -0.21, 0.71)	(0.32, -0.61, -0.31)	0.7553	Done	No
3	1	tomato	tomato	(0.14, 0.26, 0.27)	(0.34, -0.17, 0.16)	0.4124	Done	Yes
4	0	glass	cup	(-0.036, -0.21, 0.5)	(0.164, -0.4, -0.31)	0.5319	Failed	Yes
5	0	bottle	bottle	(-0.26, 0.27, 0.47)	(-0.06, -0.37, 0.17)	0.4115	Done	Yes
6	2	cup	cup	(0.37, -0.1, 0.36)	(0.57, -0.26, -0.2)	0.6576	Done	Yes
7	2	cup	cup	(-0.29, 0.16, 0.25)	(-0.09, -0.15, 0.06)	0.1849	Done	No

In the experiments other than 4, the object recognized by YOLO v3 is matched with the assigned object_id. In experiment 4, the object recognition is remarked as failed because the cup in the setup was replaced by an untrained glass which is somewhat similar in structure and colour. But still, as the YOLO v3 recognizes the trained objects based on the maximum similarity, the glass has been recognized as a cup.

Whereas in the experiments 2 and 7, the object recognized is matched with the assigned object_id but the arm couldn't reach the recognized objects. This is because the reach length of the arm is limited to 70 cm which is calculated from its 2nd joint. Hence the arm didn't reach the object in the experiment 2. Whereas in experiment 7, the arm didn't reach the object because the transformed pose value is within the self-collision avoidance region of the arm.

VII. Conclusion

This paper presented the development of a system with a robotic arm integrated with vision on the ROS platform. Kinect v2 and Kinova robotic arm have been used respectively for object recognition and manipulation purposes. The involved mathematical complexity in the system integration has been achieved using the matrix transformation method. Several tests were performed for the validation of the proposed system integration and the test results are satisfactory which have been discussed in the results section. Thus the proposed system can have effective applications in developing humanitarian technologies where vision based manipulation plays a critical role such as service robots, wheelchair, rescue robots, etc.

VIII. Future Work

To speed up the performance of YOLO v3 the system should be integrated with a high-speed processor. As a development, it will result in developing our own end effector and later integrate this into the present system. In its future, to make it user friendly the entire control of the system will be through using voice command. The robotic application can be designed to integrate with the speech application. Users can give speech commands to manipulate the objects or robotic arm motion. This technology can also be used in a wheelchair, Service robot, Rescue robots, etc. that can help physically handicapped and crippled people.

Acknowledgment

We are very thankful to Amrita Vishwa Vidyapeetham and HuT Labs for providing us all the needful lab facilities and encouragement for the successful completion of this research work. Our special thanks to our Mentor Dr. Rajesh Kannan Megalingam for his continuous guidance for this research.

REFERENCES

[1] Campeau-Lecours, Alexandre & Lamontagne, Hugo & Latour, Simon & Fauteux, Philippe & Maheu, Véronique & Boucher, François & Deguire, Charles & L'Ecuyer, Louis-Joseph. (2017). Kinova Modular Robot Arms for Service Robotics Applications. International Journal of Robotics Applications and Technologies. 5. 49-71. 10.4018/IJRAT.2017070104.

[2] https://www.who.int/news-room/fact-sheets/detail/disability-and-health

[3] https://www.who.int/publications-detail/guidelines-on-the-provision-of-manual-wheelchairs-in-less-resourced-settings

[4] R. Nakatsu, "Image/speech processing that adopts an artistic approach-toward integration of art and technology," *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Munich, 1997, pp. 207-210 vol.1.

[5] R. K. Megalingam, N. Saboo, N. Ajithkumar, S. Unny and D. Menon, "Kinect based gesture controlled Robotic arm: A research work at HuT Labs," *2013 IEEE International Conference in MOOC, Innovation and Technology in Education (MITE)*, Jaipur, 2013, pp. 294-299.

[6] W. Tang and Z. Liu, "A convenient method for tracking color-based objects in living video based on ROS and MATLAB/Simulink," *2017 2nd International Conference on Advanced Robotics and Mechatronics (ICARM)*, Hefei, 2017, pp. 724-727.

[7] Y. Hu, X. Wu, G. Zheng and X. Liu, "Object Detection of UAV for Anti-UAV Based on Improved YOLO v3," *2019 Chinese Control Conference (CCC)*, Guangzhou, China, 2019, pp. 8386-8390.

[8] H. Jeong, K. Park and Y. Ha, "Image Preprocessing for Efficient Training of YOLO Deep Learning Networks," *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*, Shanghai, 2018, pp. 635-637

[9] T. Santad, P. Silapasupphakornwong, W. Choensawat and K. Sookhanaphibarn, "Application of YOLO Deep Learning Model for Real Time Abandoned Baggage Detection," *2018 IEEE 7th Global Conference on Consumer Electronics (GCCE)*, Nara, 2018, pp. 157-158.

[10] Rajesh Kannan Megalingam, Gangireddy, R., Sriteja, G., Kashyap, A., and Ganesh, A. Sai, "Adding Intelligence to the Robotic Coconut Tree Climber", Proceedings of the International Conference on Inventive Computing and Informatics (ICICI 2017). 2017.

[11] Rajesh Kannan Megalingam, P, T., P, A., V, T., and M, G., "Robotic Arm Design for Coconut-tree Climbing Robot", International Conference on Applications and Design in Mechanical Engineering (ICADME 2015), vol. 786. pp. 328-333, 2015

[12] R. K. Megalingam, N. Katta, R. Geesala, P. K. Yadav and R. C. Rangaiah, "Keyboard-Based Control and Simulation of 6-DOF Robotic Arm Using ROS," *2018 4th International Conference on Computing Communication and Automation (ICCCA)*, Greater Noida, India, 2018, pp. 1-5.

[13] X. Zhao, Z. Cao, Q. Jia, L. Pang, Y. Yu and M. Tan, "A Vision-Based Robotic Grasping Approach under the Disturbance of Obstacles," *2018 IEEE International Conference on Mechatronics and Automation (ICMA)*, Changchun, 2018, pp. 2175-2179.

[14] Z. Zhang, J. Zheng, Y. Liu and G. Lu, "Development of a Movable Service Robot with Double Working Arms for the Elderly and the Disabled," *2009 International Conference on Electronic Computer Technology*, Macau, 2009, pp. 636-640.

[15] Ali, Md & K., Aizat & K., Yerkhan & T., Zhandos & O., Anuar. (2018). Vision-based Robot Manipulator for Industrial Applications. Procedia Computer Science. 133. 205-212. 10.1016/j.procs.2018.07.025.

[16] Vijayakumar, T. (2019). COMPARATIVE STUDY OF CAPSULE NEURAL NETWORKS IN VARIOUS APLICATIONS. Journal of Artificial Intelligence, 1(01), 19-27.