PROJECT REPORT

ON

"AUTONOMOUS ROBOT DEVELOPMENT OPEN SOURCE PLATFORM – THIRD GENERATION (ARDOP 3.0)"

Submitted in partial fulfillment of the requirements for the partial completion of

PROJECT FOR COMMUNITY SERVICE [16EC7DCPW1]

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM

Submitted by

Sudarshan S Harithas	1BM16EC109
Rajesh R	1BM16EC083

Under the guidance of

Harish V. Mekali

Assistant Professor, ECE, BMSCE

Aug – Nov 2019

Department of Electronics and Communication Engineering

B.M.S. COLLEGE OF ENGINEERING

(Autonomous College Affiliated to Visvesvaraya Technological University, Belgaum)

Bull Temple Road, Basavanagudi, Bangalore - 560019, Karnataka, India.

DECLARATION

We undersigned students of final semester B.E in Electronics and Communication Engineering, BMS College of Engineering, Bangalore, hereby declare that the dissertation entitled “Autonomous Robot Development Open Source Platform Third Generation (ARDOP 3.0), embodies the report of our project work carried out independently by us under the guidance of Mr. Harish V. Mekali, Assistant Professor, Electronics and Communication Engineering Department, BMSCE, Bangalore in partial fulfillment for the award of Bachelor of Engineering in Electronics and Communication from Visvesvaraya Technological University, Belgaum during the academic year 2019-2020.

We also declare that to the best of our knowledge and belief, this project has not been submitted for the award of any other degree on earlier occasion by any student.

Place: Bengaluru

Date: Nov 29, 2019

SUDARSHAN S HARITHAS
1BM16EC109

RAJESH R
1BM16EC083

ACKNOWLEDGMENT

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement, and cooperation of intellectuals, elders and friends. A number of personalities, in their own capacities, have helped us in carrying out this project work. We would like to take this opportunity to thank them all.

We express profound gratitude to respected principal **Dr. B. V. Ravishankar**, BMS College of Engineering for providing a congenial environment to work in.

Our sincere gratitude to **Dr. Arathi R Shankar**, Head of the Department, Electronics and Communication Engineering for encouraging and providing this opportunity to carry out the project in the department.

We would like to thank our guide **Mr. Harish V. Mekali**, Assistant Professor, Department of ECE who helped us in all the ways to carry out the project work. He stood beside and guided us in every step.

We will be incomplete, if we don’t mention our heartfelt thanks to Mr Rajath B and all the members of Propel Lab-I, BMSCE, who helped us in all the ways to carry out the project work.

We thank all our professors for providing the basic knowledge without which this project wouldn’t have been possible. Last but not least we thank our family and friends, who made their valuable support compelled us to maintain a standard throughout our endeavor.

We would like to express our profound gratitude towards team members of ARDOP2.0 Mr. Heetesh Vhavle, Mr. Akshay U Hegde , Mr. Aditya Bhat K and ARDOP1.0 Mr. Karun Warrior and Mr. Shrenik Muralidhar for their constant support.

This project would have been impossible without the support of open-source communities and platforms such as ROS Community, The Construct, Ubuntu Community, Gazebo Community, and MoveIt! Community.

Finally, we would like to thank our parents and friends for their encouragement and support.

Table of Contents

Page no

Table of Figures

1. Introduction

1.1 Abstract

Robots are becoming indispensable in today’s world. They are being used in a multitude of diff erent sectors such as surgery, autonomous driving, etc. There is a growing requirement for intelligent robots that are capable of performing tasks that are time consuming or hazardous. Robots that will have an impact in industries or consumer markets will have to integrate several domains of robotics and coordinate these eff ectively. ARDOP 3.0 (Autonomous Robot Development Open-Source Platform) is the third-generation open source humanoid robot that would serve as an extensive platform for the development of robotic applications. An economical platform aims to make robots more commonplace. The aim of this project is to build a robot capable of performing mobile manipulation by navigating through an indoor environment.

1.2 Scope

The main intention of this project is to build an intelligent autonomous robot that would be used to make acquaintance, develop physical models for implementation and forward research in the fields of computer vision, kinematics, motion planning and machine learning. The integration of these concepts is essential for any modern intelligent robot that wishes to make a mark in the rapidly advancing field of robotics. As illustrated by the Amazon Pick and Place Challenge (2016) for the development of a warehouse robot, “pick and place” forms an interesting problem definition for a robotic system and gives a body of research with scope for future betterment. ARDOP 3.0 would serve as a platform for students and researchers to implement their ideas and algorithms in this direction. They will be able to use pre-made algorithms, modify them and execute them with ease, or develop new ones and test out their executions. The diff erent abstraction layers in the project will enable people with all levels of technical expertise to build and develop algorithms and functionality in the robot. Being economical, the platform would help make robotics research more feasible.

2. Literature Survey

2.1 Vision and Perception

The object recognition problem is solved in a plethora of ways today. Computer vision based methods are moving closer into the field of machine learning in order to get more accurate results. Classifiers such as neural networks and support vector machines are widely used to classify images and they tend to work well on images containing objects. Another popular method to find shapes in an image is the Hough transform [3]. This method has been extended in many ways in order to improve performance.

The main drawback in using a traditional machine learning approach is that the features have to be manually selected. Very often, researchers spend a major portion of their time finding appropriate features that describe the dataset. The performance of the system depends heavily on the features used as the input and the model would have to be implemented and trained to evaluate the contribution of the features in producing accurate results. This is very cumbersome and time consuming process and it significantly reduces productivity. Hence, computer vision algorithms are employed in order to extract features in the images which then can be fed into a classifier. A typical image classification process is shown in

Popular method of object recognition used today is seen in [4], [5], [6]. Computer vision libraries such as OpenCV have inbuilt code written in order to implement these methods 5 quickly and they provide a real-time recognition application.

Although these algorithms work very fast and can provide high frame rates, they dont perform well on certain data sets. Although first invented in the 1960s, Convolutional neural networks have made their way back into the modern scene as computational power has increased enormously in recent times. CNNs eliminate the problem of manual feature selection by learning appropriate filters which extract the most influential features for the given data. The paper [7] marked the revival of CNNs by winning the ImageNet challenge, that year by a large margin [18].

2.2 Kinematics

Kinematics is a branch of mechanics that deals with the motion of objects without taking into account the forces that act upon them. Hence it is specifically used for pose estimation of objects-to find out their positions and orientations in three-dimensional space. Kinematics is used to study the motion of complex link-joint structures where each joint actuates a particular link system. By employing kinematic equations, it becomes possible to find out the joint parameters that would make the structure move in a way that is optimal for the given task. The end-eff ector of a robotic arm is the part that comprises of a gripper or some other manipulating structure which can be used to pick up objects, etc. The goal of the kinematic model is to enable the end-eff ector to reach the desired position to allow for interaction with real world objects. The solution obtained from the kinematic equations are the angle values for each of the joints that would make the end-eff ector move to the required position with the required orientation [8][17].

High DOF Robotic arms are typically developed based on inverse kinematics models so that the arm can reach any position or orientation in structured environments. The forward kinematic model is predicated on Denavit Hartenberg (DH) parametric scheme of robot arm position placement. Given the desired position and orientation of the robot end-eff ector, the realized inverse kinematics model provides the required corresponding joint angles.

Solving an Inverse Kinematics problem analytically means to compute the joint angles by manipulation of the Forward Kinematics equations [9]. An iterative method computes the joint angles by changing the joint angles by small amounts every iteration [10]. At one point, this process would converge and the joint angles would remain constant. The values at this point are taken to be the solution of the Inverse Kinematic problem.

2.3 ARDOP 1.0

In addition, since we are taking forward the ARDOP 1.0 [8], project and research into it is imperative. Convolutional Neural Networks (using CAFFE) is used to implement object recognition using the stereo depth map. The stereo camera used is DUO MLX, however this has a limitation that it provides only grayscale images. In order to use it with the CNNs for object recognition, RGB images are required which will improve the classification and localization accuracy. One solution would be to use additional RGB camera and requires us to solve the registration problem in order to synchronize the RGB and Depth maps. Another drawback is that the DUO MLX APIs are not open source and the camera itself is very expensive. Trajectory planning involved the computation of joint or end-eff ector trajectories to obtain required arm trajectories. Servo motors are controlled using I2C controller. Kalman Filter is used to eliminate distance measurement errors. ARDOP1.0 is shown below.

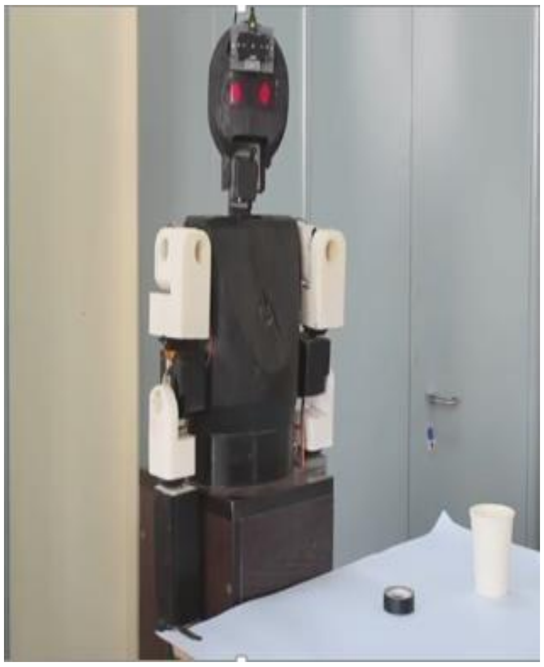


Fig 1: ARDOP1.0

2.4 ARDOP2.0

ARDOP 2.0 initiated the use of ROS and built a Moveit Package to solve the inverse kinematics in simulation. They also integrated the gazebo simulation with Rviz solution. But their design of the arm was bulky and had unintended offsets at the wrist joints this decreased the accuracy of the kinematics and increased the required torque of the motors hence increase in the amount of power sourced. They also replaced the DUO MLX with Kinect this helped in easy integration with ROS and also active use of drivers such as Openni. The motors were driven using buck and boost converters and an SMPS was used as the main power supply, this also limited the mobility of the robot. They initiated the use of Darknet and YOLO V3 for object recognition which we plan to continue for ARDOP3.0. ARDOP2.0 is shown below.

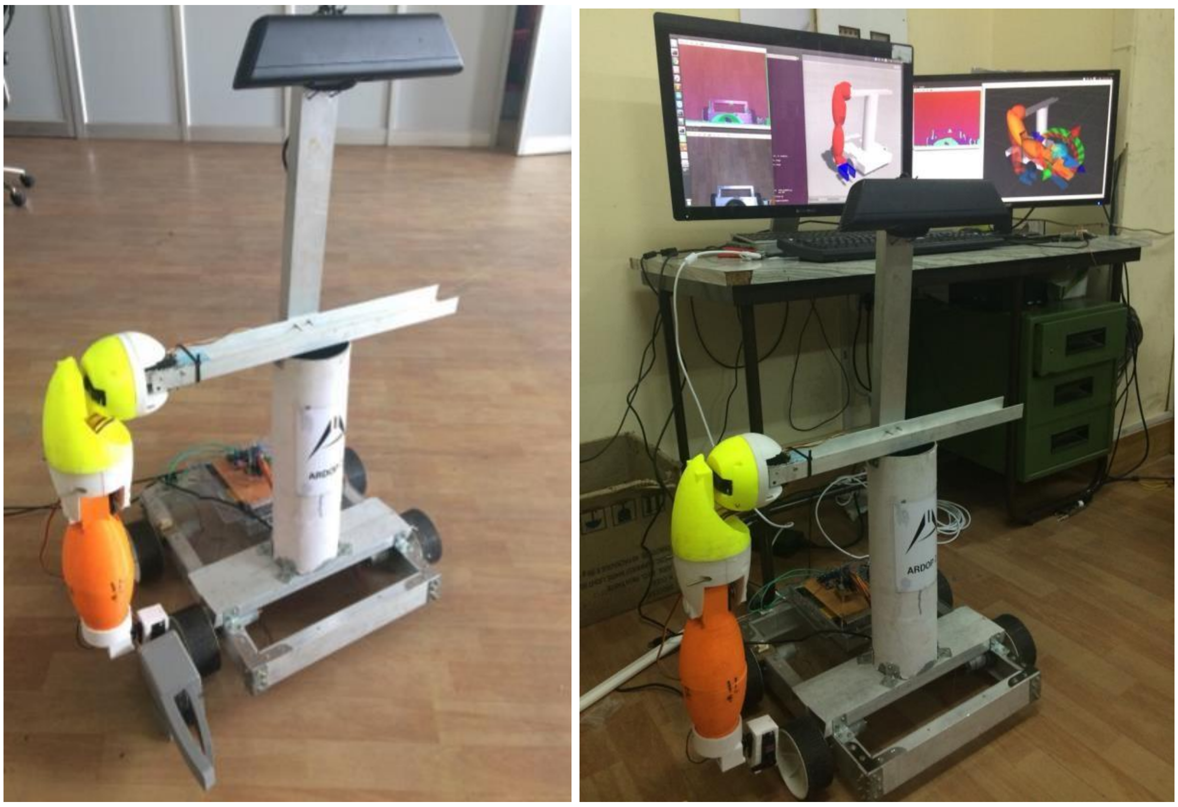


Fig 2: ARDOP2.0 with its simulation using ROS

3 Aims and Objectives

3.1 Problem Definition

The main goal of ARDOP is to develop a fully functional humanoid robot. The current aim of ARDOP 3.0 is to develop a mobile robotic arm, which will navigate to a source, pick up an object and deliver it to a destination.

There will be several objects kept in source zone. The robot will be trained to recognize these objects. We will instruct the robot to pick up one of these objects (marked in red) with the help of two 6-DOF robotic arms and a 3D camera (kinect0 for depth perception. Once picked, the robot should navigate to the destination zone, which will be across the room, and place the object in the destination. The aim of the project is shown in Fig3.

The robot will start from the starting point (O) to source zone (A) which will be the source location. There will be several objects kept in the source zone. The robot will be trained top recognized these objects. We will instruct the robot to pick up one of these objects (marked in red). Once picked, the robot should navigate to the Destination Zone (B), which will be across the room, and place the place the object in the Destination (marked in green). ARDOP 3.0 is a fully mobile robot which would be completely developed on ROS. Deep Learning models is used for object recognition

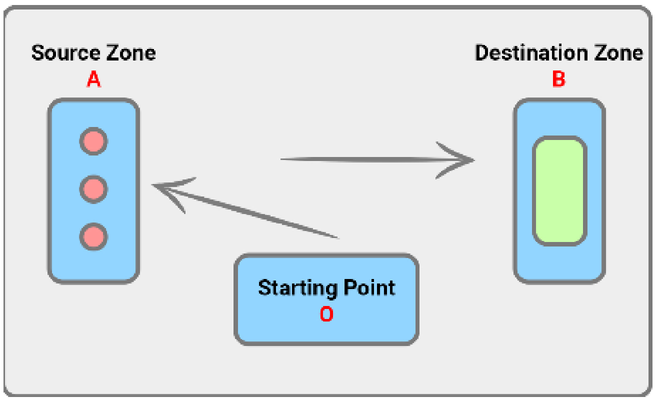


Fig3: Aim of ARDOP3.0 to perform mobile manupilation

3.2 Proposed Solution

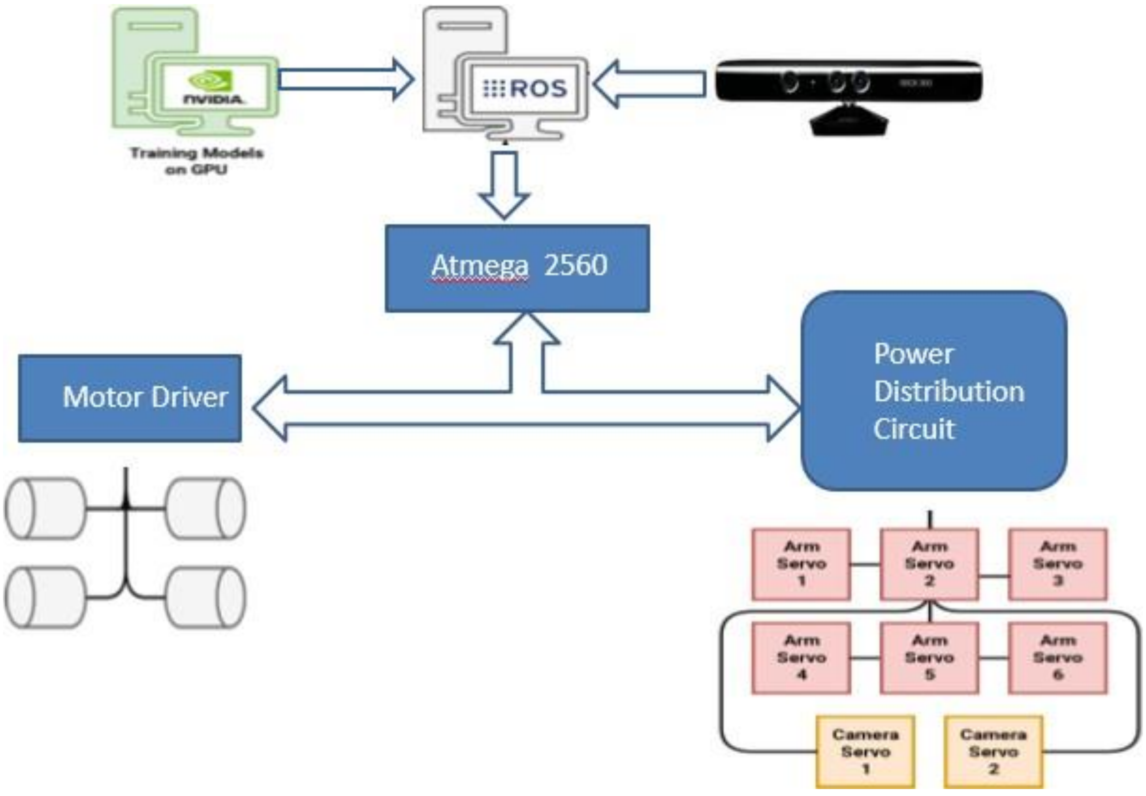


Fig4: Block diagram of ARDOP3.0

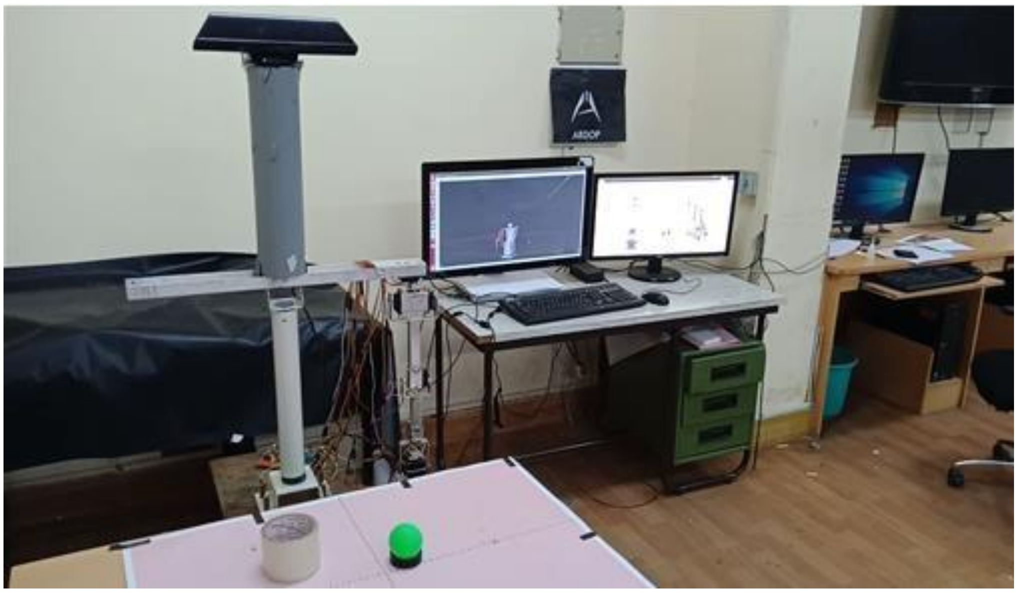


Fig5: ARDOP3.0 performing object manipulation and simulation in background

One of the primary objectives of ARDOP 2.0 is to perceive and interact with objects in its surroundings. Such a task is achieved by the seamless integration of various domains of robotics. Other open source robotics platforms do not include such a deep and extensive compilation of different concepts of intelligent robotics. Our first objective is to develop ARDOP 3.0 on ROS (Robot Operating System) which is a widely used open source platform in the robotics community for integration and scalability.

The next objective is to develop the forward and inverse kinematic models for the arms of the robot. This will allow the control of the exact position and configuration of the arm in 3D space. The entire design for the upper body of the humanoid robot is designed in SolidWorks and is 3D printed. Appropriate trajectory planning algorithms is used to ensure that the arm configurations do not interfere and collide with other objects in the scene and will be developed in MoveIt! Package of ROS. The servo motors will be controlled by Arduino Mega microcontroller via a power supply and distribution servo motor driver to control multiple servos. The entire platform is made mobile with the help of high torque Hub motors with position encoders. The chassis can support a payload of up to 150 Kg.

The final objective is to develop an object localization and classification system for the purpose of interaction. Deep Convolutional Neural Networks are used to train and classify the images of objects using Darknet and YOLO models . The networks will be initially trained on NVIDIA GPUs such as GeForce GTX 1080 Ti and 970 Ti.

3.3 Project Execution

The Design of the new arm took around three weeks. It was designed using Solid Works, The arm was made using Aluminium. In the next two weeks we got the URDF solution and also completed the simulation. Development of the designed arm and testing took another two weeks, after which we started with the testing of the arm for the next two weeks. The arm was later tested for inverse kinematics and finally the integration with the camera took two weeks.

Weeks:	1	2	3	4	5	6	7	8	9	10	11
Design of the robotic Arm	🟩	🟩	🟩								
URDF solution and Simulation				🟩	🟩						
Development of the arm and Testing						🟩	🟩				
Testing Arm for Inverese Kinematic								🟩	🟩		
Integrating with the camera										🟩	🟩

Fig 6: Project execution

4 Methodology and Implementation

The implementation of ARDOP3.0 can be broadly classified into three major domains:

- Vision and perception
- Locomotion
- Kinematics

There are multiple systems within the robot that support the working of each individual domain these include:

- ROS Simulation
- Power Supply
- Power Distribution and Servo Driver Circuit
- Mobility
- Hardware construction and integration

In this section we discuss in detail regarding the working of each section and its corresponding support system.

4.1 Vision and Perception

The Vision and Perception system are the eyes of the robot, it is the main source of information for the robot to understand its surroundings. The goals of the vision system are as follows:

- To recognize objects within a scene.
- To localize them in 3D space and extract the coordinates of the object (x,y,z)

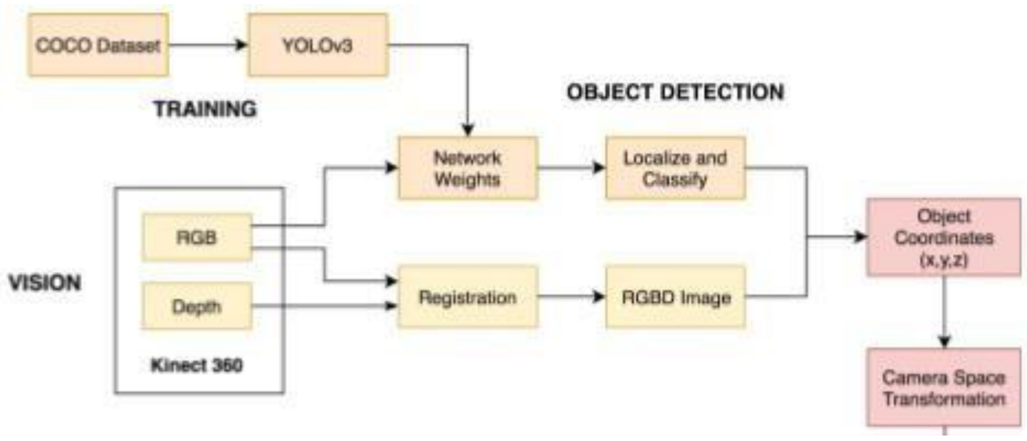


Fig7: Block diagram of Vision system

4.1.1 Object Detection and Classification

In order to pick various objects, the robot has to first figure what the object and then finds out where it is located with respect to the origin of the robot. To solve the first problem, that is classifying the object, we initially tried various classification deep learning models such as Inception models. However, this model does not give any information about the location of the object. Later models such as YOLO and SSD were explored to solve the second problem.[12]

4.1.1.1 Object Detection (Localization + Classification)

The object detection model that we are currently using is YOLOv3 (You Only Look Once). YOLO uses a single CNN network for both classification and localising the object using bounding boxes. Following is the architecture of YOLO.

YOLO divides the input image into a 13 X 13 grid. If the centre of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts five bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts. If no object exists in that cell, the confidence scores should be zero. Otherwise, we want the confidence score to equal the intersection over union (IOU) between the predicted box and the ground truth.

Each bounding box consists of 5 predictions: x, y, w, h, and confidence. The (x, y) coordinates represent the center of the box relative to the bounds of the grid cell. The width and height are predicted relative to the whole image. Finally the confidence prediction represents the IOU between the predicted box and any ground truth box. Each grid cell also predicts C conditional class probabilities, which in our case is for C=80 classes of COCO dataset. These probabilities are conditioned on the grid cell containing an object. YOLO only predict one set of class probabilities per grid cell, regardless of the number of boxes. Finally, the box confidence score and classification probabilities are multiplied to get the overall object detection score.

The main advantages of YOLO over classification models are as follows:

- Provides bounding box data for objects, which is required to localize the objects.
- YOLO can detect multiple objects in the same frame along with the confidence levels for classifying each of the objects.

- Speed – YOLOv3 runs at 45 fps, which is better than real-time. However, on the current system we are using, YOLO runs only at 2.5 fps.

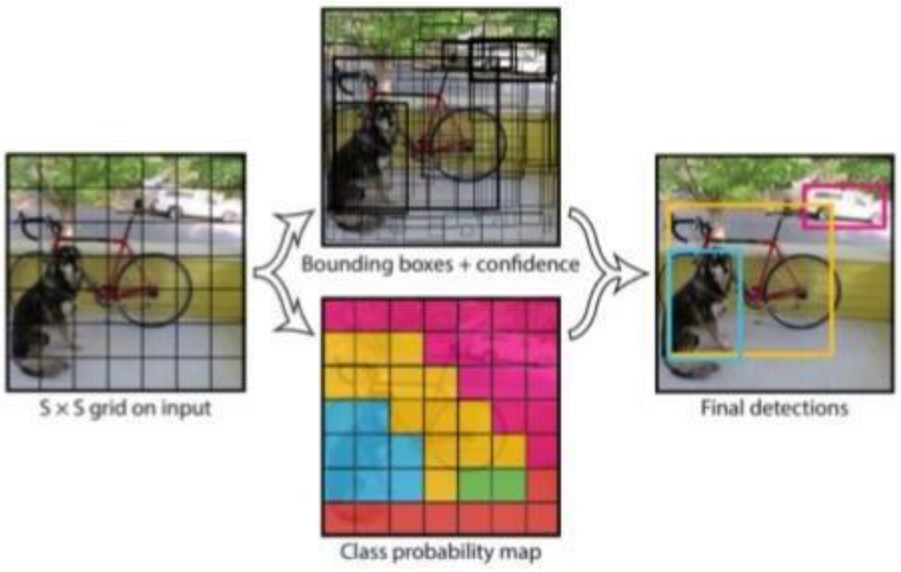


Fig 8: YOLO Prediction Flow diagram.

Source: https://towardsdatascience.com/ yolo-you-only-look-once-real-time-object-detection-explained

4.1.2 Object Localization in 3D space

Openni is a driver that is used with ROS and it publishes the topic '/camera/depth/image_raw" this topic provides the distance in millilitres from the plane of the camera to the point in each pixel. This value is used and passed to the coordinate transform function which transforms it to the camera base frame.

4.2 Locomotion

The Mobility platform consist of a chassis with 4 wheels that was previously driven using a DC motor of 60Kg-cm torque and a speed of 30rpm with quadrature encoder. Currently we shifting towards using Hub motors instead of DC, as it provides an added advantage of increasing the load on the chassis and makes it suitable for outdoor use. The mobility platform has been built using Aluminium bars and the base is made of wood. The motors are housed in the chassis.

Navigation of the robot has the aim of providing specific locations in space within Gazebo, to which the robot attempts to move. Classically, within ROS, the navigation stack is used to achieve navigation of a robot provided certain prerequisites are met.

The hardware requirements to be met for the robot to be compatible with the navigation stack are as follows:

- It is meant for both diff erential drive and holonomic wheeled robots only. It assumes that the mobile base is controlled by sending desired velocity commands to achieve in the form of: x velocity, y velocity, theta velocity.
- It requires a planar laser mounted somewhere on the mobile base. This laser is used for map building and localization.
- The Navigation Stack was developed on a square robot, so its performance will be best on robots that are nearly square or circular. It does work on robots of arbitrary shapes and sizes, but it may have difficulty with large rectangular robots in narrow spaces like doorways.

The prerequisites for the software stack of the robot are as follows:

- The robot must be running ROS.
- It must have a tf transform tree in place
- It must publish sensor data using the correct ROS Message types.
- The Navigation Stack needs to be configured for the shape and dynamics of a robot to perform at a high level.

4.3 Kinematics

Kinematics deals with techniques required to move the robotic arm to a desired position in space. Kinematics can be broadly classified in forward and inverse kinematics. Forward kinematics is used to determine the position and orientation of the end effector in space given the joint angles, inverse kinematics is used to solve the problem of determining the joint angles of the arm, given the end effector position and orientation. [13]

4.3.1 Forward Kinematics

Forward Kinematics of the robotic arm calculates the position of the endeff ector for given joint variables. The complexity of the kinematic equations increases with increase in joints. Also, increase in the Degrees of Freedom (DoF) of the arm makes it difficult to compute the mathematics. Let us derive the forward kinematics for the ARDOP3.0 arm, it consists of two 6DOF robotic arms inclusive of a spherical wrist which enables it to grasp objects in all possible orientations. Forward kinematics is obtained using the Denavit Hartenberg (DH) technique.

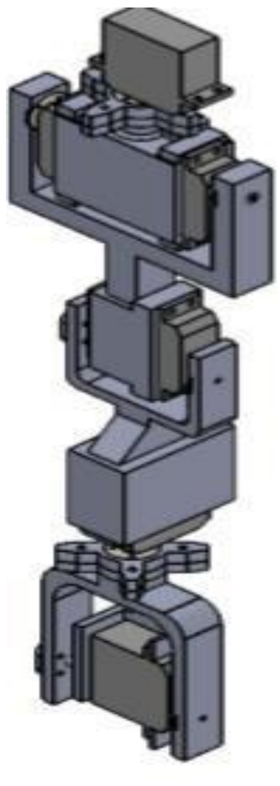


Fig 9: 6DOF arm design

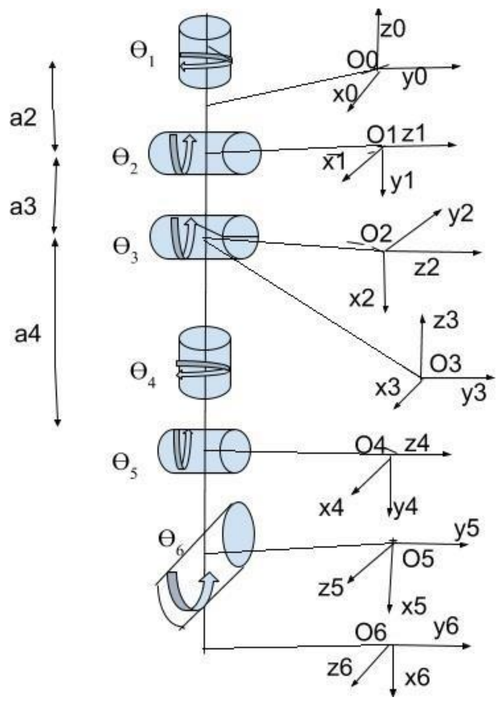


Fig 10: Coordinate frames of the arm

a2	4 cm
a3	19.5 cm
a4	33 cm

	Θ_0	α	r	d
1	Θ_1	-90	0	a_2
2	Θ_2+90	0	a_3	0
3	Θ_3-90	90	0	0
4	Θ_4	-90	0	a_4
5	Θ_5-90	-90	0	0
6	Θ_6	0	0	0

Table 1: DH parameter table

$$Hn+1 = \begin{pmatrix} \cos(\Theta_n) & -\sin(\Theta_n)\, coa(\alpha_n) & \sin(\Theta_n)\cos(\alpha_n) & r_n\cos(\Theta_n) \\ \sin(\Theta_n) & \cos(\Theta_n)\cos(\alpha_n) & -\cos(\Theta_n)\sin(\alpha_n) & r_n\sin(\Theta_n) \\ 0 & \sin(\alpha_n) & \cos(\alpha_n) & d_n \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$H_n^{n+1} = H(\Theta_n, \alpha_n, r_n, d_n)$$

$$H_0^6 = H_0^1.H_1^2.H_2^3.H_3^4.H_4^5.H_5^6$$

The variables d, a, θ, α are all defined as per DH conventions which are as follows.

- d: off set along previous z to the common normal.
- θ: angle about previous z, from old x to new x.
- a: length of the common normal.
- α: angle about common normal, from old z axis to new z axis.

For a good 6 DOF arm design, three of the joints should be such that it helps in making the end-eff ector reach the spacial point and the rest three joints should facilitate in setting the orientation of the end-eff ector.

Forward Kinematics using the Geometry based Technique: Geometry is an effective tool that can be utilized to derive the forward kinematics solution. The only drawback is that the geometry based technique only accounts for the spatial location of the end effector but not its orientation. The equations that govern the spatial location of the end effector are:

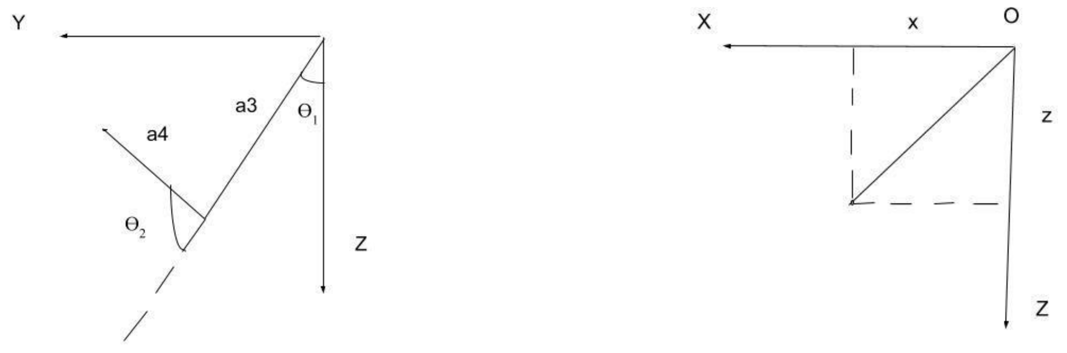


Fig11: FK using geometry based technique

$$L=a_3\sin(\Theta_1) + a_4\sin(\Theta_1+\Theta_2)$$

$$x=L\sin(\Theta_0)$$

$$y=L\cos(\Theta_0)$$

$$z= a_3\cos(\Theta_1) + a_4\cos(\Theta_1+\Theta_2)$$

4.3.2 Inverse Kinematics

Once the forward kinematics is done and equations of the arm is found the inverse kinematics is needed to be computed. For given spatial coordinates in order for the arm to move, the values of the joint angles have to calculate in order for it to reach that position. The mathematical complexity of inverse kinematics is higher than that of the forward kinematics. The reason for this is that there can be multiple solutions for one position and also possible that there is no solution at all.

Here we implement inverse kinematics using two techniques:

- Inverse Kinematics using Geometry
- Inverse Kinematics using the Jacobian Pseudo Inverse method

4.3.2.1 Inverse Kinematics using Geometry

For inverse kinematics we would know the end effector location in space (x,y,z) and we are expected to determine the joint angles corresponding to that location and orientation. [9]

The angle Θ_0 can be directly obtained using the equation shown below

$$\Theta_0=\tan^{-1}(X/Y)$$

Pythagoras Theorem can be used to obtain the effective distance in 2D plane

$$R_2=\sqrt{x^2+y^2}$$

$$r_1=z$$

$$\Phi_2=\tan^{-1}(r_2/\ r_1)$$

$$R_3=\sqrt{r_1{}^2+r_2{}^2}$$

Using Cosine formula we can obtain Φ_1

$$\Phi_1=\cos^{-1}\left(\frac{a_3{}^2+r_3{}^2-a_4{}^2}{2a_3a_4}\right)$$

Using the above results we can obtain the value for Θ_1

$$\Theta_1=\Phi_1-\Phi_2$$

$$\Phi_3=\cos^{-1}\left(\frac{a_3{}^2+a_4{}^2-r_3{}^2}{2a_3a_4}\right)$$

$$\Theta_2=180^\circ-\Phi_3$$

If the geometry based method is used to determine the angles Θ_0 , Θ_1, , Θ_2 these values can be substituted into the DH matrix up to H_0^3 and multiply the inverse of this matrix with the end effector coordinates and equate it with H_3^6 and obtain the value of Θ_3 , Θ_4, , Θ_5

4.3.2.2 Inverse Kinematics using the Jacobian Pseudo Inverse method

The following algorithm is used to implement the inverse kinematics using the Jacobian Pseudo Inverse Method. [10]

- The initial θ value is initialized randomly. Iter ← 0.
- For i=1 to 500 do steps 3-5
- The Jacobian and its pseudoinverse is computed using this θ
- The input value for the target position is used to compute Δt.
- The theta update is calculated and theta is updated.
- The end eff ector position is calculated using the updated θ.
- The error between the required target and the calculated target is computed.
- If the error is less than the threshold and the calculated angles are valid, the solution has been found and the program ends.
- Iter ← Iter + 1. If Iter > 500, No solution is possible. Else, go to step 1.

$X=f(q)$

$X=[x, y, z]$

$q=[\Theta_0, \Theta_1, \Theta_2, \Theta_3, \Theta_4, \Theta_5]$

Mathematical Expressions for the above algorithm

$$\Delta X_n = X_d - X_n$$

$$\Delta q_n = J(q_n)^{-1} \Delta X_n$$

$$q_{n+1} = q_n + \Upsilon (q_n)$$

Loop through the above lines “n” times.

$$J(q) = \frac{\partial f}{\partial q} = \begin{pmatrix} \frac{\partial f1}{\partial q1} & \cdots & \frac{\partial f1}{\partial qn} \\ \vdots & \ddots & \vdots \\ \frac{\partial fm}{\partial qm} & \cdots & \frac{\partial fm}{\partial qn} \end{pmatrix}$$

To compute the Jacobian Matrix:

$$L=a_3\sin(\Theta_1) + a_4\sin(\Theta_1+\Theta_2)$$

$$x=L\sin(\Theta_0)$$

$$y=L\cos(\Theta_0)$$

$$z= a_3\cos(\Theta_1) + a_4\cos(\Theta_1+\Theta_2)$$

$$\frac{\partial x}{\partial \Theta_0} = Lcos(\Theta_0)$$

$$\frac{\partial x}{\partial \Theta_1} = [a_3 \cos(\Theta_1) + a_4 \cos(\Theta_1 + \Theta_2)]\sin(\Theta_0)$$

$$\frac{\partial x}{\partial \Theta_2} = [a_4 \cos(\Theta_1 + \Theta_2)]\cos(\Theta_0)$$

$$\frac{\partial y}{\partial \Theta_0} = Lsin(\Theta_0)$$

$$\frac{\partial y}{\partial \Theta_1} = [a_3 \cos(\Theta_1) + a_4 \cos(\Theta_1 + \Theta_2)]\cos(\Theta_0)$$

$$\frac{\partial y}{\partial \Theta_2} = [a_4 \cos(\Theta_1 + \Theta_2)]\cos(\Theta_0)$$

$$\frac{\partial z}{\partial \Theta_0} = 0$$

$$\frac{\partial Z}{\partial \Theta_1} = [-a_3 \sin(\Theta_1) - a_4 \sin(\Theta_1 + \Theta_2)]$$

$$\frac{\partial Z}{\partial \Theta_2} = -a_4 \sin(\Theta_1 + \Theta_2)$$

$$J(q) = \begin{pmatrix} \frac{\partial x}{\partial \Theta_0} & \frac{\partial x}{\partial \Theta_1} & \frac{\partial x}{\partial \Theta_2} \\ \frac{\partial y}{\partial \Theta_0} & \frac{\partial y}{\partial \Theta_1} & \frac{\partial y}{\partial \Theta_2} \\ \frac{\partial z}{\partial \Theta_0} & \frac{\partial z}{\partial \Theta_1} & \frac{\partial z}{\partial \Theta_2} \end{pmatrix}$$

4.3.3. Kinematics of the head joint: Coordinate Transformation

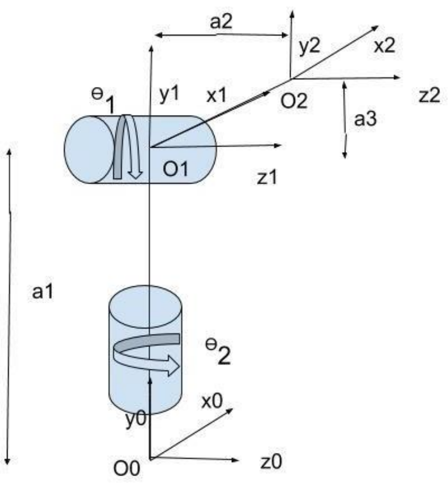


Fig12: Coordinate frames of the head and its transform to base frame

	Θ_0	α	r	d
1	$90+\Theta_1$	90	0	a_1
2	Θ_2	0	0	0

Table 2: Parameter table of the head frame

a1	4 cm

$$Hn+1 = \begin{pmatrix} \cos(\Theta_n) & -\sin(\Theta_n)\,coa(\alpha_n) & \sin(\Theta_n)\cos(\alpha_n) & r_n\cos(\Theta_n) \\ \sin(\Theta_n) & \cos(\Theta_n)\cos(\alpha_n) & -\cos(\Theta_n)\sin(\alpha_n) & r_n\sin(\Theta_n) \\ 0 & \sin(\alpha_n) & \cos(\alpha_n) & d_n \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

4.3.4 Error Prediction

This is an extremely vital step in increasing the accuracy of the kinematics, the new arm had an issue with its mechanical construction the error was dependent on the input angle . In order to compensate for this error it was necessary to predict the error beforehand. Hence a database comprising of the input angle and its corresponding error was created, a graph was plotted for these values and a fourth order polynomial trend line was drawn. The graph is shown below for each input angle the corresponding error is determined and used in the program. The equation was determined to be:

$$-1.96 \times 10^{-6} x^{4} + 3.57 \times 10^{-4} x^{3} - 0.0231x^{2} + 0.751x - 0.756 = 0$$

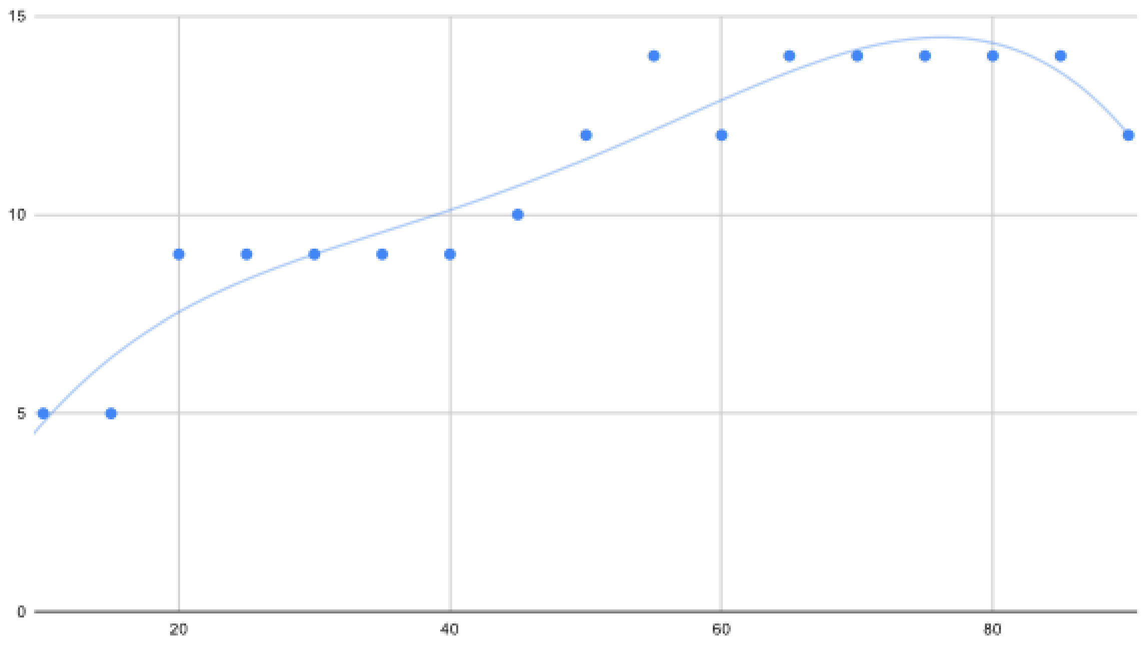


Fig13: The input angle vs Expected error plot

4.3.5 Simulation of the Robot Using ROS:

Robot Operating System or ROS is an open source flexible framework used for the development of robotic softwares. Even though ROS is not an Operating System it acts as a robotic middleware for heterogeneous computer cluster such as message-passing between processes, implementation of commonly used functionality, hardware abstraction, low-level device control, and package management. It is a set of libraries, tools, and conventions that facilitates in simplifying the complexity of mathematics and mechanics required to design the robot. It also servers as a common platform to many robots in the world.

ROS supports various planner softwares and simulators like MoveIt!, Rviz, and Gazebo for various robotic applications like arm motion planner, navigation, environment mapping, camera calibration, point cloud mapping, etc. ROS by itself is not a real-time framework but it possible to integrate ROS with real-time code. ROS supports libraries development through roscpp and rospy.

4.3.5.1 Moveit!

MoveIt! is state of the art software for mobile manipulation, incorporating the latest advances in motion planning, manipulation, 3D perception, kinematics, control and navigation [14]. It provides an easy-to-use platform for developing advanced robotics applications, evaluating new robot designs and building integrated robotics products for industrial, commercial, R&D and other domains.

MoveIt! uses various motion planners such as OMPL(Open Motion Planning Library), Stochastic Trajectory Optimization for Motion Planning (STOMP), and Covariant Hamiltonian Optimization for Motion Planning (CHOMP), and supports Kinematics plugins like KDL (Kinematics and Dynamics Library), IKFast, etc. All the motion planners and plugins can be integrated with MoveIt! through move group node.

MoveIt! RViz Plugin: MoveIt! Comes with a plugin for the ROS Visualizer (RViz). The plugin allows to setup scenes in which the robot will work, generate plans, visualize the output and interact directly with a visualized robot. Once the robot description (URDF and SRDF) are generated, the robot is launched into RViz to help visualise and interact with the robot.

MoveIt! Inverse Kinematics: MoveIt! Supports various kinematics plugin for the inverse kinematic solution. There are default kinematics plugin available in MoveIt!. This can be chosen during MoveIt! Setup Assistant. However, it can later be edited from configuration file.

KDL — Kinematics and Dynamics Library is an open source package developed by Orocos Project. It is a meta-package that depends on orocos kdl which contains the C++ version and python orocos kdl which contains the generated python bindings. KDL includes excellent support to work with vectors, points, frame transformations, etc. It can calculate a vector product, transform a point into a diff erent reference frame, or even change the reference point of a 6d twist. It is also possible to represent a kinematic chain by a KDL Chain object, and use KDL solvers to compute anything from forward position kinematics, to inverse dynamics.

The forward kinematics is given by

$$\dot{X} = J\dot{\theta}$$

Where X is the position matrix and θ is the joint angle matrix. KDL solves these equations by using linear least square method i.e.

$$\left\| \dot{X} - J\dot{\theta} \right\|$$

In order to solve this least square problem, Newton-Raphson method is used. This gives an approximate solution.

4.3.5.2 Gazebo

Gazebo is a simulator to simulate the robot before actually implementing it on hardware to prevent any catastrophic damages to the robot. Gazebo has its own engine to compute the physics for the robot world. Gazebo ros pkgs is a set of ROS packages that provide the necessary interfaces to simulate a robot in the Gazebo 3D rigid body simulator for robots. It integrates with ROS using ROS messages, services and dynamic reconfigure. In order to use Gazebo in simulating the robot there has to be 'Gazebo tags' in the robot description URDF file.

4.6 Power Distribution and Servo Driver Circuit

The previous generations of ARDOP had been powered using a SMPS it converts the AC supply to regulated and filtered DC supply and can source a maximum of 40A of current at 12V. ARDOP3.0 is mobile robot and hence a SMPS cannot be deployed. A 12V 60Ah , lead acid battery has been used as the source of power for the entire robot. This battery is capable of driving the entire robot which consist 16 servo motors, 4 DC motors and a Kinect.

Lead acid, Lithium ion and polymer, Nickel cadmium batteries, were the options that we had explored. Lead acid type secondary battery has been selected because of it's because of its heavy duty performance and cost effective as compared to other batteries.

Various components of a robot work at different power rating, the DC motors draw a peak of 7A of current at 12V, the servos work between voltage ranging between 5-8.5V and current between 1- 3.5A. To satisfy this diverse power requirements a power distribution and Servo driver circuit was built using buck and boost dc converters. These converters have the following specification:

Fig 14: DC-DC buck and Boost converter

- Input voltage: 7-40V.
- Output voltage: 1.2 – 35V, adjusted using on-board potentiometer.
- Output current: 0-8A, adjusted using on-board potentiometer.
- Output power(max) : 300W

5 Results

5.1 Vision and Perception

All the objectives of vision and perception were achieved successfully. Object detection using YOLOv3 was integrated and the network can classify and localize 80 diff erent objects of the COCO dataset. The detection results are shown below. The vison system can localize objects within 0.2 cm of error. The results of object recognition is shown below in Fig 14.

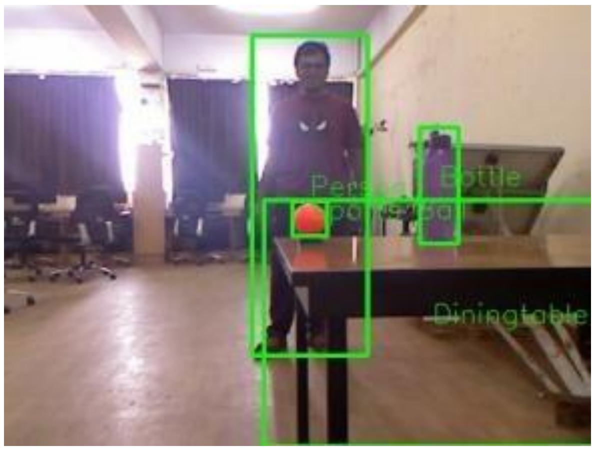


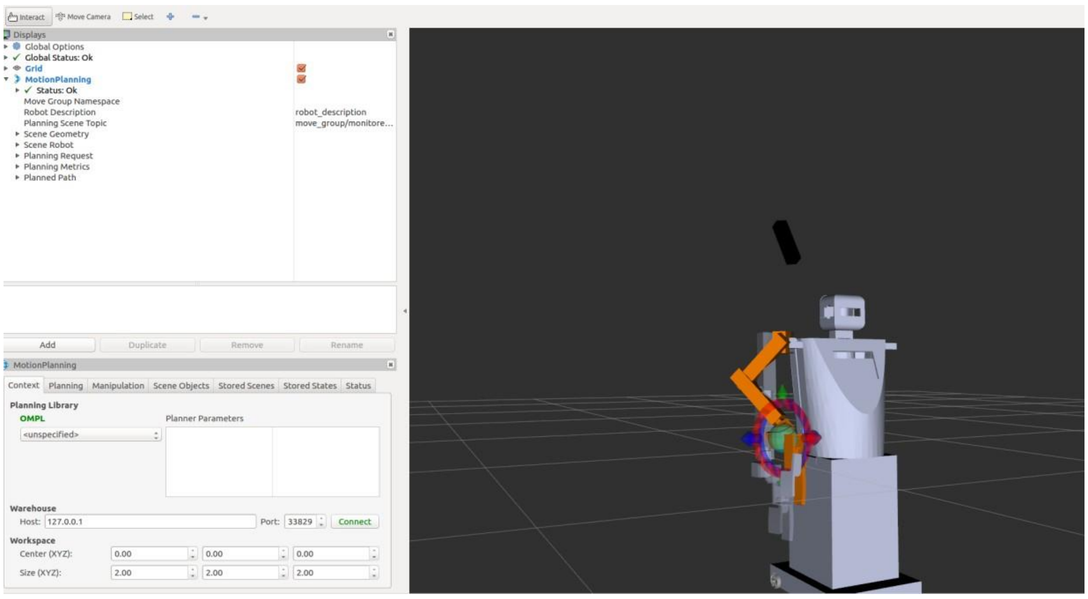
Fig 15: YOLO V3 results

5.2 Kinematics and Manipulation

- **Simulation Results:**

The simulation was started by testing the URDF of the robot that was developed using solidworks. This model was imported into ROS, and a Moveit Package was built. Moveit consist of IK solvers (Inverse Kinematics) such as KDL (Kinematics and Dynamics Library) , using these libraries the IK was solved. Obstacle avoidance was also employed using collision aware IK Fig 17 and 18.

Rviz is a visualizer and Gazebo is the real world representation of the robot in simulation. The arm is moved to the desired position using the available interactive markers or a python script that passes the coordinates of the final destination can be used. Once the coordinates and the orientation of the end effector is known KDL is used to develop the IK and trajectory plan. The execution of the visualizer (Rviz) can be seen in the real world model (Gazebo) Fig 15 and 16.



```
Global Options
Global Status: Ok
Grid
MotionPlanning
  Status: Ok
  Move Group Namespace
  Robot Description          robot_description
  Planning Scene Topic       move_group/monitore...
  Scene Geometry
  Scene Robot
  Planning Request
  Planning Metrics
  Planned Path

Planning Library
  OMPL                       Planner Parameters
  <unspecified>

Warehouse
  Host: 127.0.0.1                    Port: 33829

Workspace
  Center (XYZ):   0.00   0.00   0.00
  Size (XYZ):     2.00   2.00   2.00
```

Fig 16: The destination and orientation of end effector has been set using interactive markers

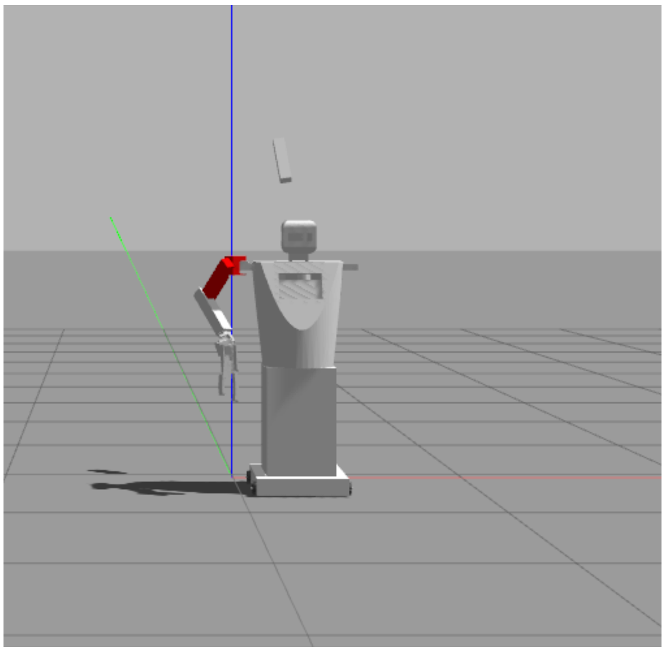
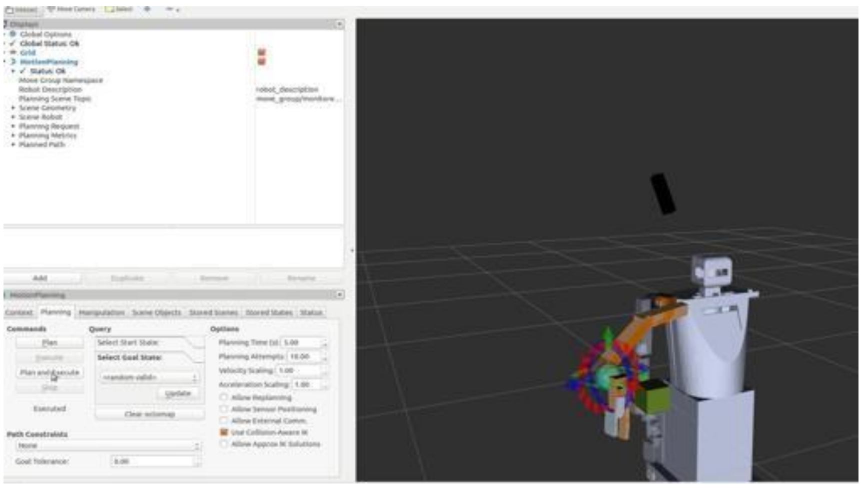
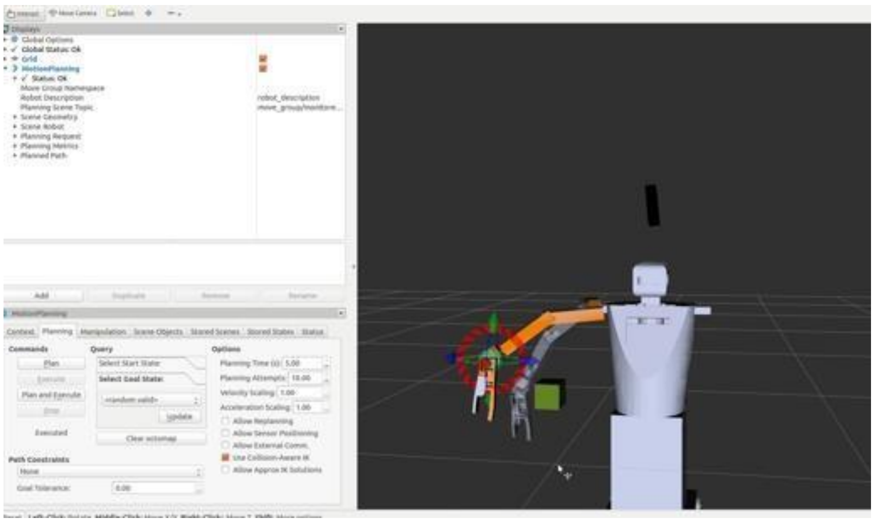


Fig17: The plan visualizsed in Rviz is being executed in the simulation and end effector is movd to desired poisiton.

The robot is capable of avoiding obstacles within the workspace of the arm, this has been simulated in Rviz and a collision aware IK has been developed.



Planning Time (s): 5.00
Planning Attempts: 10.00
Velocity Scaling: 1.00
Acceleration Scaling: 1.00
☐ Allow Replanning
☐ Allow Sensor Positioning
☐ Allow External Comm.
☑ Use Collision-Aware IK
☐ Allow Approx IK Solutions
Path Constraints: None
Goal Tolerance: 0.00



Planning Time (s): 5.00
Planning Attempts: 10.00
Velocity Scaling: 1.00
Acceleration Scaling: 1.00
☐ Allow Replanning
☐ Allow Sensor Positioning
☐ Allow External Comm.
☑ Use Collision-Aware IK
☐ Allow Approx IK Solutions
Path Constraints: None
Goal Tolerance: 0.00

Fig 18 and 19: In both of the above figures collision avoidance is dmonstraed the IK is genreated and the trajectory is planned such that the obstacle (Cube) , is avoided.

A robot with two arms has been built and a Moveit package has been generated, the simulation results are show below Fig 19, 20, 21.

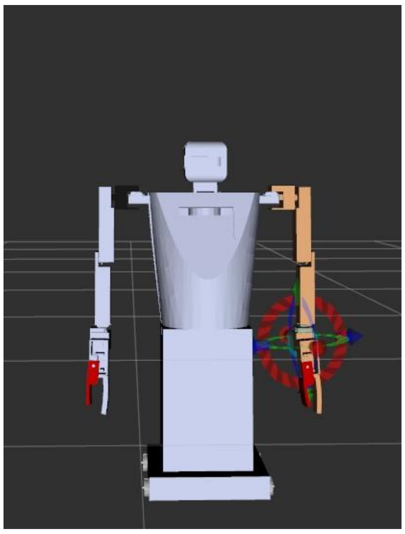
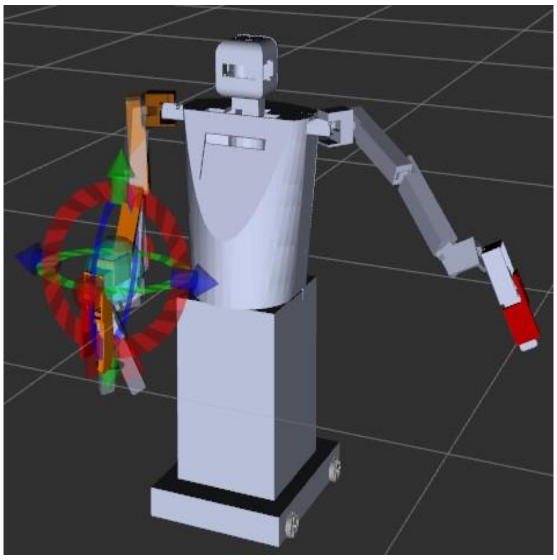


Fig 20: ARDOP3.0 simulation with two arms



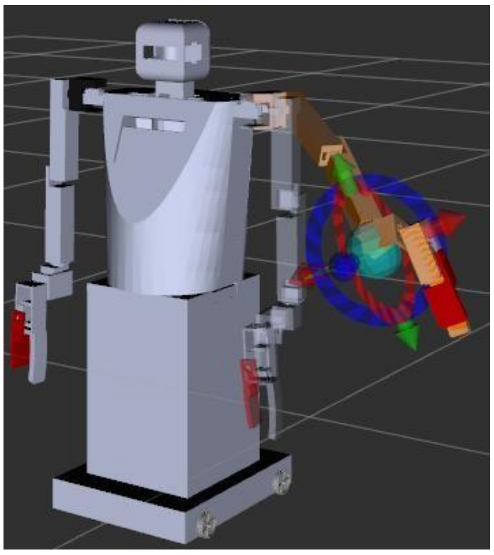


Fig 21 and 22: Setting goal pose of right and left arm using interactive markers

- **Hardware Implementation Results**

During the kinematics testing, the motors were given various angles and the theoretical and practical values were observed and the error was determined. The obtained results is presented below.

θ_0	θ_1	θ_2	Theoretical			Practical		
			x	y	z	x	Y	z
0.0	30.0	30.0	0.0	38.57	33.52	0.0	38.0	34.5
0.0	30.0	40.0	0.0	41.0	28.6	0.0	41.0	29.5
0.0	30.0	60.0	0.0	43.0	17.32	0.0	43.5	18.0
0.0	30.0	0.0	0.0	26.5	45.89	0.0	27.0	44.5
0.0	60.0	0.0	0.0	45.46	26.25	0.0	45.0	26.0
0.0	50.0	0.0	0.0	41.0	34.06	0.0	41.0	34.0
0.0	70.0	0.0	0.0	49.8	18.2	0.0	50.0	19.0
0.0	45.0	0.0	0.0	37.47	37.47	0.0	38.0	37.0
10	30	30	6.65	37.74	33.38	6.6	38.0	33.5
20	30	40	13.94	38.30	28.17	14	28.3	38.5
30	30	60	21.37	37.02	16.88	21.4	37.0	17.0
40	30	0	16.87	22.73	45.46	16.9	23.0	45.5
50	45	45	35.84	30.73	13.78	35.9	31.00	13.9

The errors are less than: $x < 0.06$ cm; $y < 0.5$cm; $z < 1$ cm

A trajectory plan was developed such that the robot picks up the object and drops it into the cup without colliding with any of the obstacles such as a table in this case Fig 22, 23, 24.

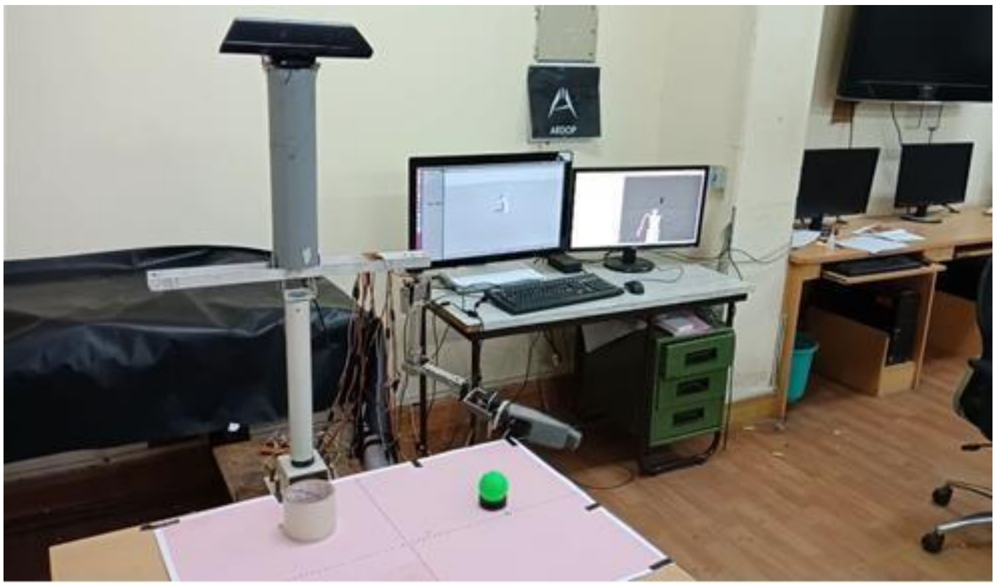


Fig 23: Avoiding the obstacle Table , simulation can be seen in the background

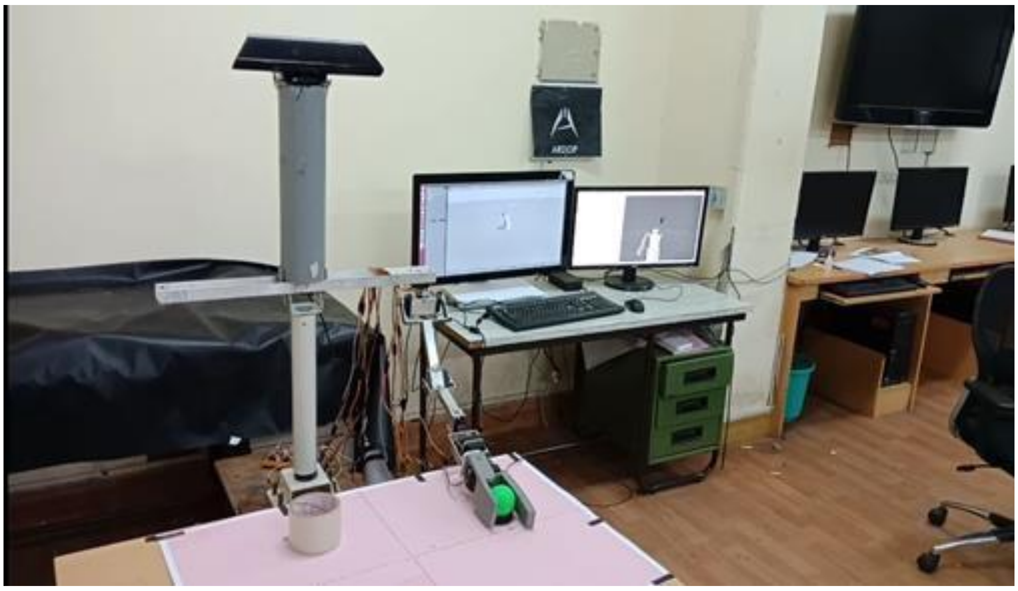


Fig24: Pick operation

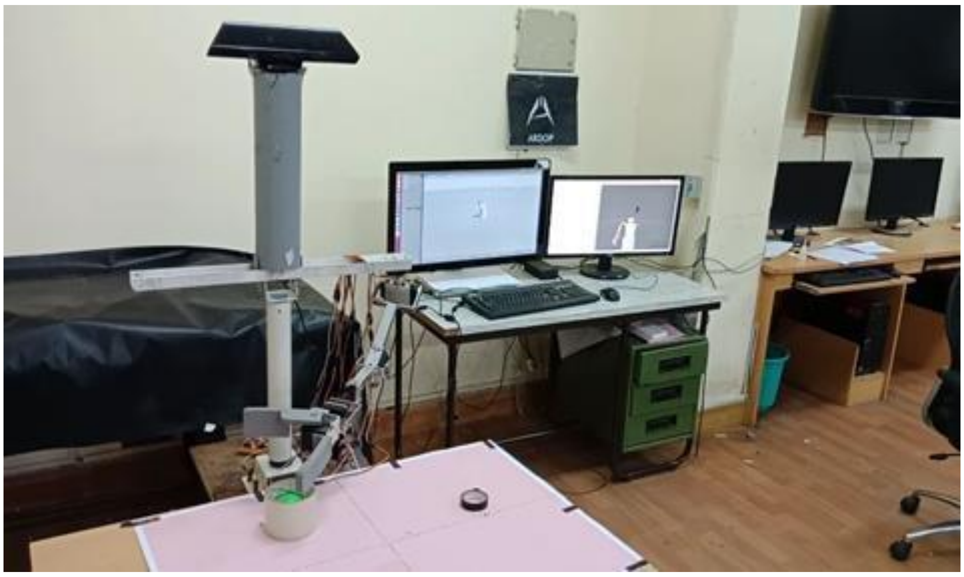


Fig25: Place Operation

6 Conclusion

We have successfully achieved the integration of the vision and the kinematic system, the arm works functions as per expectations with an error of: $x < 0.06$ cm; $y < 0.5$cm; $z < 1$ cm this error can be caused on course of measurements. The camera is able to localize objects within an error of 0.2 cm.

The simulation of the robot has been successfully completed by generating Moveit package, the simulation has also been extended to ARDOP3.0 with two arms.

7 Future Work

Our next task would be to construct a similar arm and make this robot autonomous, mobile by integrating SLAM with the vision and kinematic system and use a suitable laptop or Jetson Tx1/Tx2/Tk1 (Nvidia) for on board processing. The design is shown below.

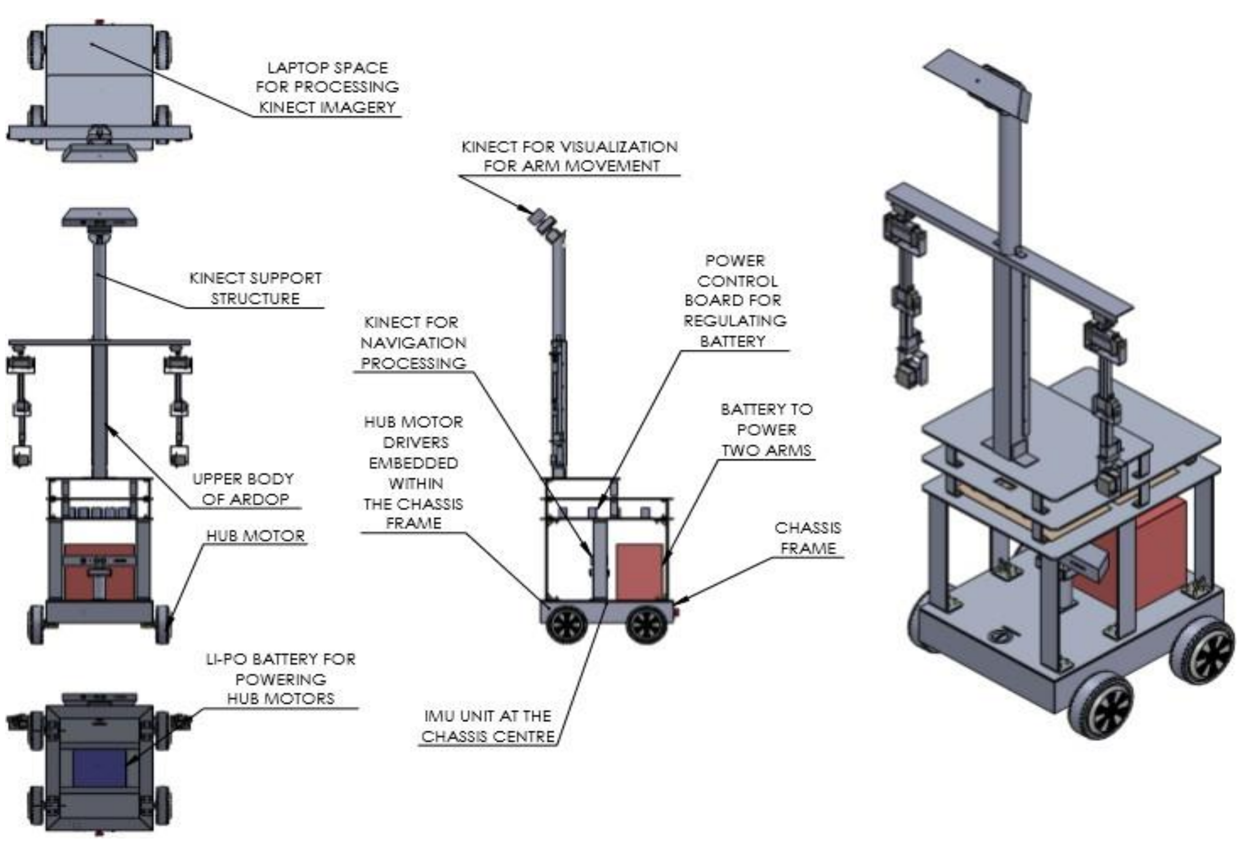

LAPTOP SPACE FOR PROCESSING KINECT IMAGERY
KINECT FOR VISUALIZATION FOR ARM MOVEMENT
KINECT SUPPORT STRUCTURE
POWER CONTROL BOARD FOR REGULATING BATTERY
KINECT FOR NAVIGATION PROCESSING
HUB MOTOR DRIVERS EMBEDDED WITHIN THE CHASSIS FRAME
BATTERY TO POWER TWO ARMS
UPPER BODY OF ARDOP
HUB MOTOR
CHASSIS FRAME
LI-PO BATTERY FOR POWERING HUB MOTORS
IMU UNIT AT THE CHASSIS CENTRE

Fig 26: The future development of ARDOP3.0 with SLAM implementation

References

[1] Staff (Sandia National Labs) (August 16, 2012), Lifelike, cost eff ective robotic hand can disable IEDs, R and D Magazine, rdmag.com, retrieved September 13, 2012

[2] IEEE Xplore: The Canadarm grasps this boom and can position it in the necessary positions to permit a complete inspection.

[3] John Illingworth and Josef Kittler. A survey of the hough transform. Computer vision, graphics, and image processing, 44(1):87116, 1988.

[4] David G Lowe. Object recognition from local scale-invariant features. In Computer vision, 1999. The proceedings of the seventh IEEE international conference on, volume 2, pages 11501157. Ieee, 1999.

[5] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, volume 1, pages 886893. IEEE, 2005.

[6] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. Computer visionECCV 2006, pages 404417, 2006

[7] Alex Krizhevsky, Ilya Sutskever, and Geoff rey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 10971105, 2012.

[8] Karun Warrior, Shrenik Muralidhar, Autonomous Robot Development Open Source Platform, BMS College of Engineering, 2017.

[9] Chris Welman. Inverse kinematics and geometric constraints for articulated figure manipulation. Simon Fraser University, 1994.

[10] Andreas Aristidou and Joan Lasenby. Inverse kinematics: a review of existing techniques and introduction of a new fast iterative solver. University of Cambridge, Department of Engineering, 2009.

[11] G.J. Iddan & G. Yahav, 3D IMAGING IN THE STUDIO (AND ELSEWHERE...), 3DV Systems Ltd., Yokneam, Israel.

[12] Redmon, Joseph and Farhadi, Ali. YOLOv3: An Incremental Improvement, arXiv, 2018.

[13] Reza N. Jazar Theory of Applied Robotics: Kinematics, Dynamics, and Control Second Edition, Springers.

[14] Khokar, Karan & Beeson, Patrick & Burridge, Robert. Implementation of KDL Inverse Kinematics Routine on the Atlas Humanoid Robot. Procedia Computer Science, Massachusetts, 46. 1441-1448. 10.1016/j.procs.2015.02.063.

[15] S. Macfarlane, E.A. Croft, IEEE Transactions on Robotics and Automation 19(1), 42 (2003)

[16] S.R. Buss, IEEE Journal of Robotics and Automation 17(1-19), 16 (2004)

[17] R. Jazar, Theory of Applied Robotics: Kinematics, Dynamics, and Control (Springer, 2010). URL https://books.google.co.in/books?id=ESTJAfpf9zwC

[18] S. Suzuki, et al., Computer vision, graphics, and image processing 30(1), 32 (1985)

[19] J. Canny, IEEE Transactions on pattern analysis and machine intelligence (6), 679 (1986)

[20] G. Welch, G. Bishop, (1995)

[21] A. Gasparetto, V. Zanotto, Mechanism and machine theory 42(4), 455 (2007)

[22] D. Xu, C.A.A. Calderon, J.Q. Gan, H. Hu, M. Tan, International Journal of Automation and Computing 2(2), 114 (2005)