International Journal of
Advances in Scientific Research and Engineering (ijasre)

E-ISSN : 2454-8006

DOI: 10.31695/IJASRE.2021.34020

Volume 7 (6) 2021
June - 2021

Modelling, Simulation, and Implementing ROS for Autonomous Navigation of Tracked Robot

Wessam Hussein[1], **Shaimaa M. Maged**[2], **El-Sayed Adel**[3], **Sherif Sabry**[4], **Omar Abubaker**[5], **Mazen Amr Mustafa**[6], **Ahmed Essam Dakroury**[7], and **Essam Morsy**[1]

Department of Mechanical Engineering

Canadian International College

Cairo, Egypt

ABSTRACT

This paper presents autonomous navigation and its implementation based on Robotic Operating System (ROS) for a non-holonomic autonomous tracked robot, whereas it handles a full design, implementation, dynamic modeling, and kinematic modeling of the robot. This paper applies robot localization using Adaptive Monte Carlo Localization (AMCL) which uses a particle filter to track the pose of a robot against a known map. It also uses mapping based on the mapping package. It provides laser-based SLAM (Simultaneous Localization and Mapping) mapping. The LIDAR is used to create a 2-D occupancy grid map. Path planning is established using the Dijkstra algorithm to plan a path to a goal position, using a persistent map created by the robot during the mapping process. Also, the robot was equipped with a camera module for image processing and detection of different objects for security purposes. The proposed system shows its capability for adaptation to road segments of different curvatures and the transitions between them.

Key Words*: Autonomous Vehicle, Dynamic Model, Kinematic Model, Path Planning, Image Processing,*

1. INTRODUCTION

Autonomous navigated vehicles have the potential to reduce human risks during disaster response as well as the associated costs. They are required to effectively support a rescue squad performing high-level tasks such as exploring the disaster area, detecting and rescue victims that is why building and maintaining a persistent representation of the environment, accurate localization, and efficient path planning are fundamental prerequisites [14]. This developed system is also applicable for several industrial and military applications such as transportation, agricultural, medical, and inspection [16,18,10,5,4]. Therefore, they have to avoid all obstacles that they might encounter, yet still, drive through the tightest spaces that they can physically fit. However, our Autonomous navigated tracked robot is also designed to inhabit unmodified office-like environments that are shared with people. By relying on a planar, obstacle sensor, and laser rangefinder, because vertical structure dominates man-made environments, these sensors are positioned on the robot to detect obstacles along with a horizontal slice of the world. The result is a robot that can easily avoid walls, chair legs, feet, etc.

In this paper, we present a robot navigation system that exploits three-dimensional obstacle data to avoid the smallest obstacles that can be perceived with the available sensors yet drive through any spaces that the robot can fit. The controller has been implemented as a ROS package providing scalability and exportability to the system to be used with a wide variety of simulators and real vehicles. For obstacle detection, an RPLIDAR A1 is integrated on our tracked robot for many Operations.

2. TRACKED ROBOT STRUCTURE AND DESIGN

The structure of a model should be both light and strong. In addition, the robot will be capable to operate in different environments. In a lot of situations, it may be inconvenient, dangerous, or impossible to have a human operator present on the side which makes solving those situations very difficult or even impossible to solve. The mass of the model plays an important role in its design, it must be light to provide easier control and to use less power motor and give a longer life battery. therefore, aluminium is selected for the robot because of its lightweight.

2.1 Tracked Robot Computer-Aided Design (CAD) Model

CAD model is drawn using SOLID-WORKS version 2020. the whole assembly of robot parts is described, as shown in figure (1).

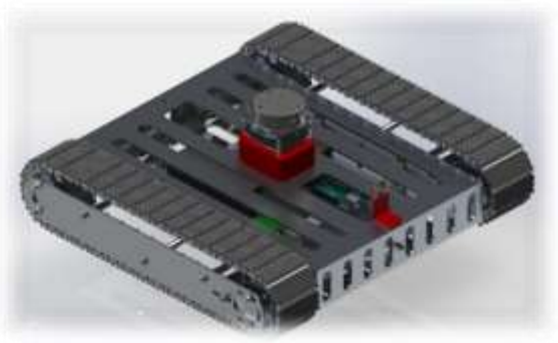


Figure 1: The Tracked Robot Assembly Drawing

As the exploded view shows the whole structure of the robot consisting of chassis that is used as a housing for the control unit and power unit. The differential drive system which is used to move the robot consists of two dc motors that transfer the rotary motion via coupling to the two sprockets which synchronize the motion to the belt to track the robot. Fixation of the whole structure used socket head cap screw, as shown in figure (2).

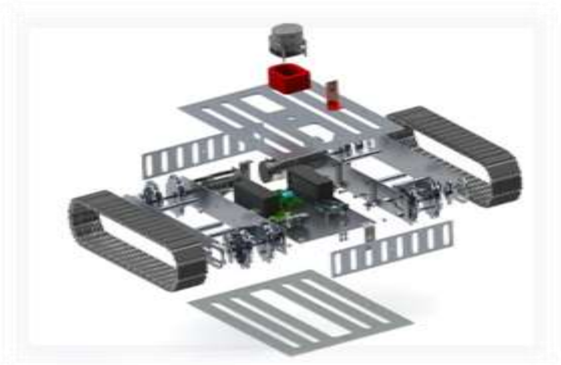


Figure 2:The Parts of The Assembly Drawing

2.2 Finite Element Analysis (FEA):

Finite element analysis was made to ensure design safety, As FEA enables you to predict potential design issues and therefore minimize risk [2].

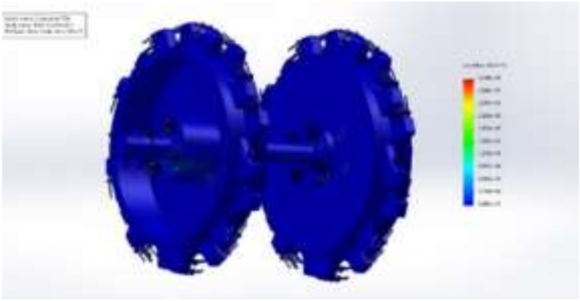


Figure 3:Motor shaft coupling to sprockets from the motor side

EMG 49 motor shaft is coupled to a sprocket that drives the robot track, the motor stall torque is about 10.5 N.m, it is assumed in the above analysis that motor torque is slightly higher (15 N.m) and the torque required to rotate the Sprockets from a static position is about 1 N.m, it is assumed in the analysis to be about 5 N.m.

Figure 4: Motor shaft coupling to sprockets from flanged bearing side

3. DYNAMIC MODEL

This unit describes the dynamic model of a tracked vehicle as a non-holonomic robot that has a 3-DOF which allows the robot to move in the longitudinal and lateral direction. The robot is designed with two rear-drive motors, both of them are driven with two dc geared motors individually, where the total torque of the motors is transmitted to the other sprockets of the tracked robot and also goes to the other wheels according to a synchronous belt that located on the sprockets.

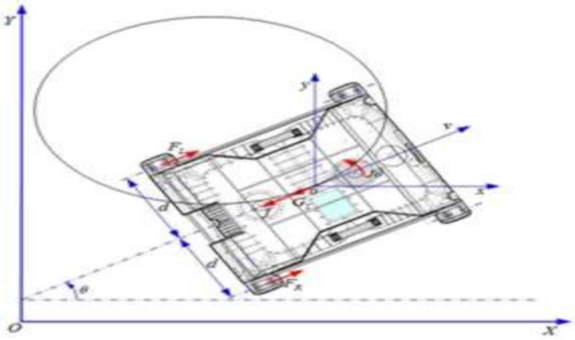


Figure 5:a free body diagram of a tracked vehicle

As shown on the graph a free body diagram of a tracked vehicle moving on a horizontal plane in a global frame of reference XOY. According to newton second law of motion the equation of force and moment can be expressed as:

$$\sum F = ma$$

$$\sum F_x = m\ddot{x}$$

$$\sum F_y = m\ddot{y} \qquad (1)$$

$$\Sigma M_\theta = I_z\ddot{\theta}$$

Lagrange dynamic method is a very influential technique for expressing the equations of motion of any mechanical structure. This method, which was introduced by Lagrange, is used to systematically derive the equations of motion by considering the kinetic and potential energies of the given system [1]. The equation of motion for the robot according to Lagrange and newton Euler methodology, is derived as:

$$\ddot{\theta} = \left(\frac{L}{Md^2+J}\right)\left(\frac{\tau_r - \tau_L}{R}\right) - \frac{Md}{Md^2+J}\left(\dot{\theta}_u\right) \qquad (2)$$

(angular acceleration)

$$\dot{u}_u = d\dot{\theta}^2 + \frac{(\tau_r + \tau_L)}{MR} \qquad (3)$$

(transitional velocity)

DOI: 10.31695/IJASRE.2021.34020

3.1 MATLAB\Simulink

By building the block diagram of the robot dynamics according to the equations of motion derived by Lagrange and newton Euler approach:

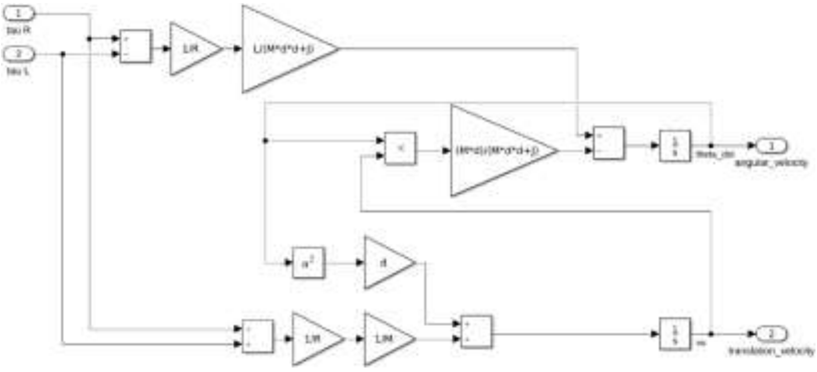
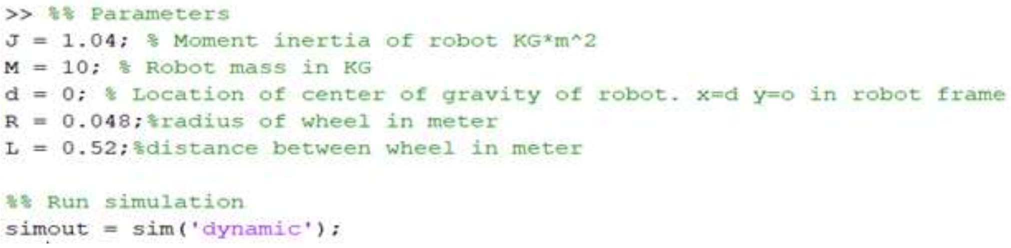


Figure 6: Block Diagram describing angular and transitional velocity.

As the input of the system will be the torque produced from the right and the lift motors (τ_r and τ_L). The output of the system is the translation speed ($\dot{u}_u$)and the angular speed ($\dot{\theta}$).

3.2 Simulation Results

Results of simulation for the motion response of the robot according to the parameters of the tracked robot:



```
>> %% Parameters
J = 1.04; % Moment inertia of robot KG*m^2
M = 10; % Robot mass in KG
d = 0; % Location of center of gravity of robot. x=d y=o in robot frame
R = 0.048;%radius of wheel in meter
L = 0.52;%distance between wheel in meter

%% Run simulation
simout = sim('dynamic');
```

Figure 7: parameters of the tracked robot

If the input torque for the robot for $\tau_r = 5\ N.m\ = \tau_l$ both will take the same value: (the robot will move forward only without any slip)

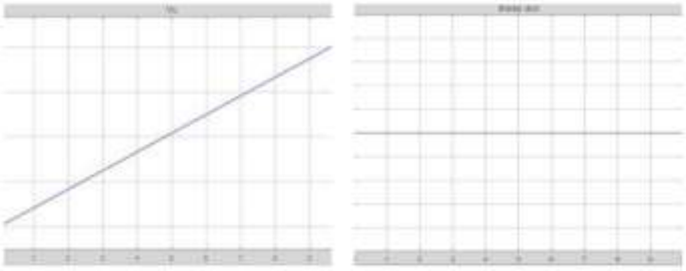


Figure 8:simulation of linear and angular velocity_ first case

In the second case if the robot took inputs as $\tau_r = 5\ N.m$, $\tau_l = 0\ N.m$: (the robot will move in the left direction, there is no transitional motion).

DOI: 10.31695/IJASRE.2021.34020

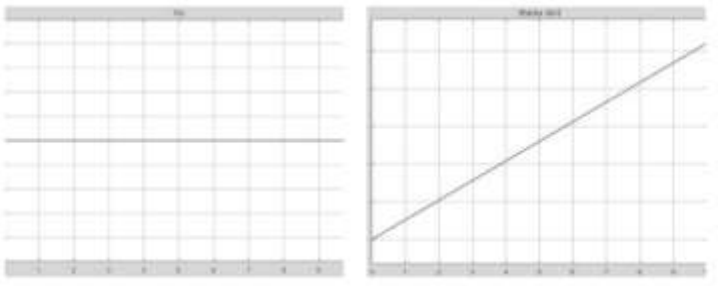


Figure 9:simulation of linear and angular velocity _second case

3.3 Tractive Force

The maximum tractive force developed by a track is determined by the area A of the tracks in contact with the terrain, and the maximum shear strength of the terrain τ_m can be expressed by Coulomb's equation [2]:

$$F_m = A\tau_m = A(C + P\tan\phi) = AC + \frac{W}{2}\tan\emptyset \tag{4}$$

Where; $\boldsymbol{A = bl}$, where track width and $\boldsymbol{b}$ track ground contact length $\boldsymbol{l}$. represent the apparent soil cohesion, $\boldsymbol{[N / m^2]}$, $\emptyset$ the angle of internal shearing resistance (friction) of the terrain or soil, [deg].

$\boldsymbol{P}$ being the normal pressure beneath the track, $\boldsymbol{[N / m^2\ ]}$, $\boldsymbol{W}$ the weight of the vehicle. **[N]** For a specified track slip σ_i at a given instant, the tractive force of a track is calculated as [17]:

$$F_i = F_m\left[1 - \frac{k}{\sigma_i l}\left(1 - e^{\frac{-\sigma_i l}{k}}\right)\right] \tag{5}$$

in which k represents the soil shear deformation modulus with units of $\boldsymbol{[m]}$.

4. Kinematic Model

Many tracked robots use a drive mechanism known as differential drive. It consists of two sprockets mounted on a common axis, each sprocket is separately being driven either forward or backward by a DC Motor and each sprocket drives its belt. And we can change the velocity of each sprocket, to perform the rotation motion, the robot must rotate about a certain point that lies along their common left and right sprocket axis. The point that the robot rotates about is known as ICC - Instantaneous Centre of Curvature. According to Dudek and Jenkin, Computational Principles of Mobile Robotics [7].

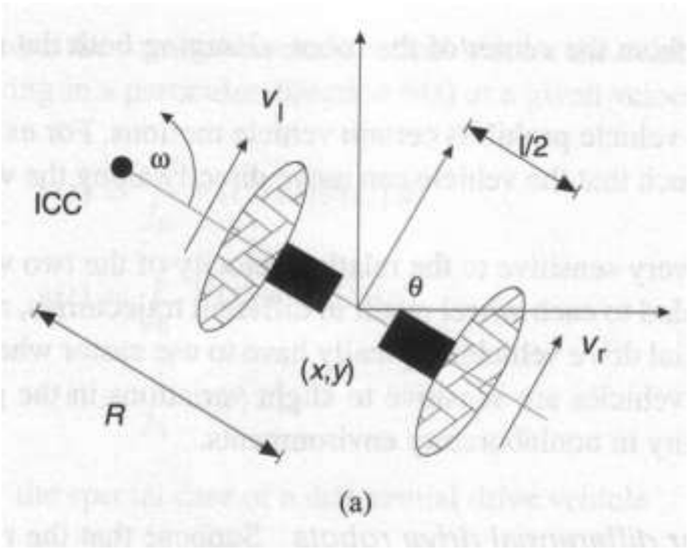


Figure 10: Differential Drive kinematics

DOI: 10.31695/IJASRE.2021.34020

By changing the velocities of the two sprockets, we can change the trajectories that the robot takes. Because the rate of rotation ω about the ICC must be the same for both wheels, we can write the following equations:

$$\omega\left(R+\frac{l}{2}\right)=V_r \tag{6}$$

$$\omega\left(R-\frac{l}{2}\right)=V_l \tag{7}$$

where Vr , Vl are the right and left sprockets velocities along the ground, l is the distance between the centers of the two sprockets, and R is the signed distance from the ICC to the midpoint between the wheels. At any instance in time, we can solve for R and ω:

$$R=\frac{l}{2}\frac{V_l+V_r}{V_r-V_l} \quad ; \quad \omega=\frac{V_r-V_l}{l} \tag{8}$$

(1) If $Vl = Vr,$ then we have forward linear motion in a straight line. R becomes infinite, and there is effectively no rotation - ω is zero.

(2) If $Vl = -Vr$, then R = 0, then rotation is about the midpoint of the sprocket axis, therefore the robot rotates in place.

(3) If $Vl = 0$, then rotation is about the left sprocket. In this case R =$\frac{l}{2}$. Same is true if $Vr = 0$.

5. Hardware Description and ROS-Integration

5.1 Platform Description

As shown in figure (11) the research platform is a small, tracked robot is driven by EMG49 24\ V DC motor with wheel encoders. These motors are operated by a power amplifier MD49 Dual 24 Volt and 5 Amp motor drive H-Bridge was used as a motor controllers/ driver. Whereas MD49 motor driver/ controller communicates with the Arduino-UNO controller via UART communication protocol, and the Arduino takes the signals from Raspberry-Pi 4 board via USB serial, and Arduino is also powered by USB port.

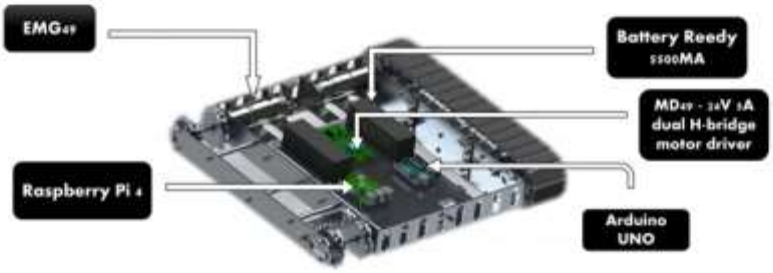
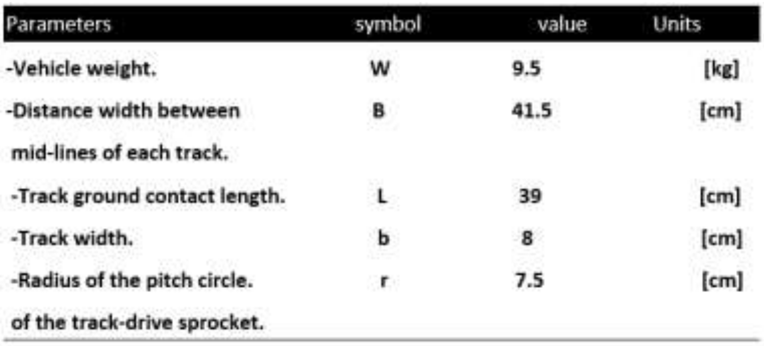


Figure 11:The robot with the system components.

Table 1 Tracked robot and terrain physical parameters.



Parameters	symbol	value	Units
-Vehicle weight.	W	9.5	[kg]
-Distance width between mid-lines of each track.	B	41.5	[cm]
-Track ground contact length.	L	39	[cm]
-Track width.	b	8	[cm]
-Radius of the pitch circle. of the track-drive sprocket.	r	7.5	[cm]

Table 1: System Parameters

DOI: 10.31695/IJASRE.2021.34020

5.2 ROS Implementation

In ROS implementation a laptop machine Lenovo-y720 [Intel(R) Core (TM) i7-7700HQ CPU @ 2.80GHz] With 16GB Ram and works as a master computer operated with Linux (Ubuntu) version 18.04 operating system. There is a slave machine that works with the master machine which is Raspberry Pi 4B running Ubuntu-mate 18.04 [13]. As shown in figure (12).

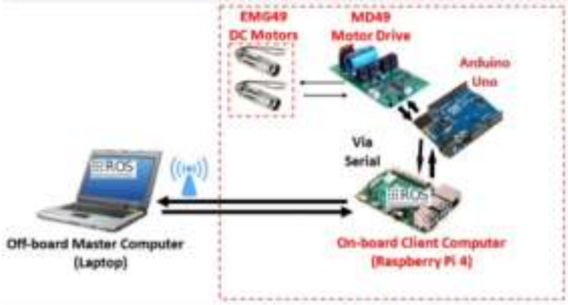


Figure 12:system architecture block diagram of the autonomous

Robot operating system (ROS) is Linux-based, is a flexible framework for creating robot software.it consists of tools, libraries, and protocols that aim to simplify and creating complex robot behavior across numerous of robotics platform [13]. ROS structure consists of packages and nodes that are used to build a complete workspace for the robot, in order to build the strategy that is followed to build a robot in ROS is to create Nodes to connect to the controller to ROS environment, while the components, mapping, localization and move_ base nodes are implemented by C++ and python shared libraries that get linked to the node at compiling time.

ROS structure allows the tracked robot to link and synchronize messages between nodes in the master machine and with Raspberry-pi 4 board on board client computer and micro-controller in the robot by the network using roscore master computer.

For nodes contacts to each other according to topics and that collected and described through ROS graph diagram, that shows in figure (13). It displays several nodes and topics to control tracked robot autonomous systems. Graph of ROS nodes came in shape of (ellipses) and topics (squares) of the proposed tracked vehicle. The continuous line arrows are topic subscriptions, that moved from subscriber node to publish it into another node with directions going from the subscriber node to the publisher one.

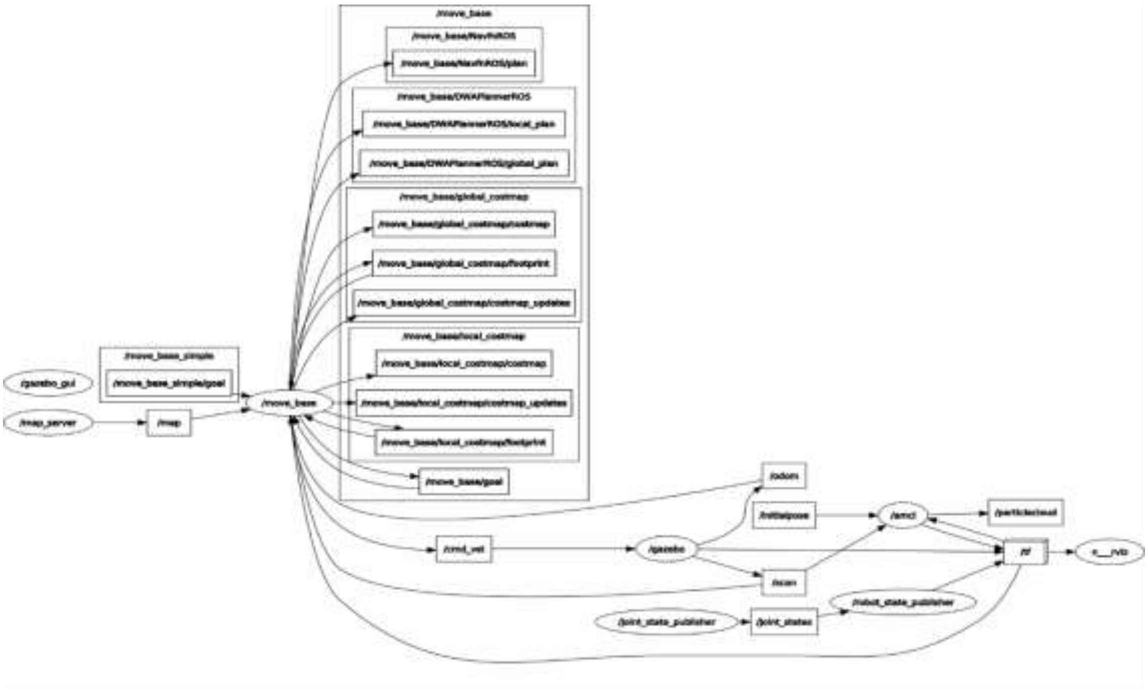


Figure 13: ROS Computation graph diagram rqt graph

For links and joints of the tracked robot autonomous system it is created by Unified Robot Description format **URDF** that described in ROS transformation tree frames diagram It shows all the frames involved in the architecture of the system, that shown in figure (14).

DOI: 10.31695/IJASRE.2021.34020

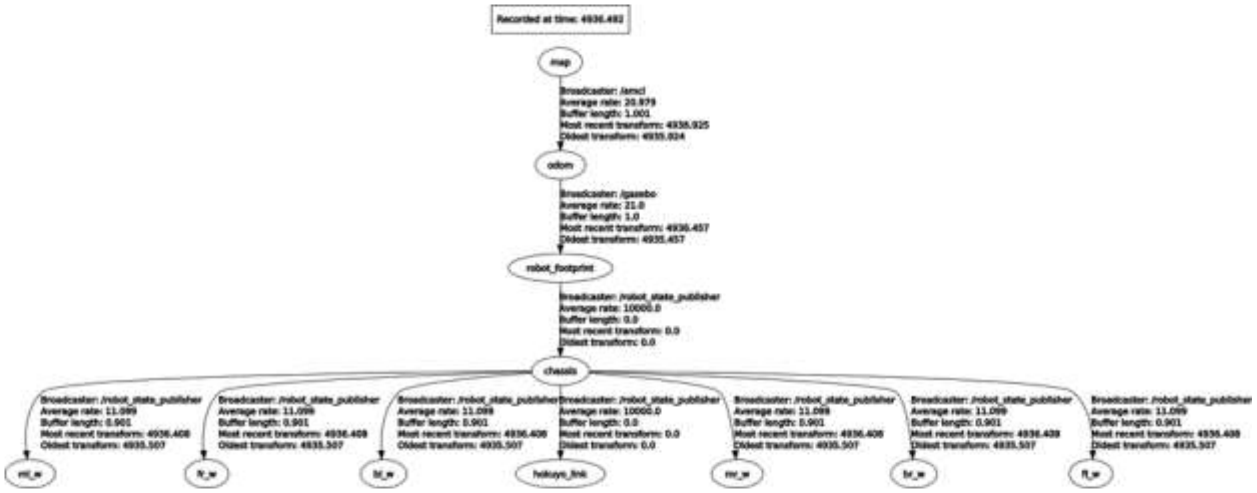

Recorded at time: 4936.492
map
Broadcaster: /amcl
Average rate: 20.979
Buffer length: 1.001
Most recent transform: 4936.925
Oldest transform: 4935.924
odom
Broadcaster: /gazebo
Average rate: 21.0
Buffer length: 1.0
Most recent transform: 4936.457
Oldest transform: 4935.457
robot_footprint
Broadcaster: /robot_state_publisher
Average rate: 10000.0
Buffer length: 0.0
Most recent transform: 0.0
Oldest transform: 0.0
chassis
Broadcaster: /robot_state_publisher
Average rate: 11.099
Buffer length: 0.901
Most recent transform: 4936.408
Oldest transform: 4935.507
Broadcaster: /robot_state_publisher
Average rate: 11.099
Buffer length: 0.901
Most recent transform: 4936.409
Oldest transform: 4935.507
Broadcaster: /robot_state_publisher
Average rate: 11.099
Buffer length: 0.901
Most recent transform: 4936.408
Oldest transform: 4935.507
Broadcaster: /robot_state_publisher
Average rate: 10000.0
Buffer length: 0.0
Most recent transform: 0.0
Oldest transform: 0.0
Broadcaster: /robot_state_publisher
Average rate: 11.099
Buffer length: 0.901
Most recent transform: 4936.408
Oldest transform: 4935.507
Broadcaster: /robot_state_publisher
Average rate: 11.099
Buffer length: 0.901
Most recent transform: 4936.409
Oldest transform: 4935.507
Broadcaster: /robot_state_publisher
Average rate: 11.099
Buffer length: 0.901
Most recent transform: 4936.408
Oldest transform: 4935.507
ml_w
fr_w
bl_w
hokuyo_link
mr_w
br_w
fl_w

Figure 14:ROS transformation tree frames diagram rqt tf tree

5.3 URDF in Gazebo

The Unified Robotic Description Format (URDF) is an XML file format used in ROS to define all parts of a robot. To use a URDF file in Gazebo, some additional simulation-specific tags must be added to work properly with Gazebo. This section describes the essential steps to successfully use your URDF-based robot in Gazebo, saving you from having to generate a separate SDF file from scratch and duplicating description formats. Under the hood, Gazebo will then convert the URDF to SDF automatically [8,13]. The first step to getting your robot working in Gazebo is to have a working URDF file therefore we used the approach of creating a cad design using Solidworks cad drawing program and then converting the CAD model into URDF format by URDF EXPORTER and then put it in an XML file format. If the URDF is created and suitable as a standard format for gazebo it will be displayed in gazebo as shown in figure(15), Therefore, after the transformation of CAD design of URDF file format handled and set by using visual studio code program.

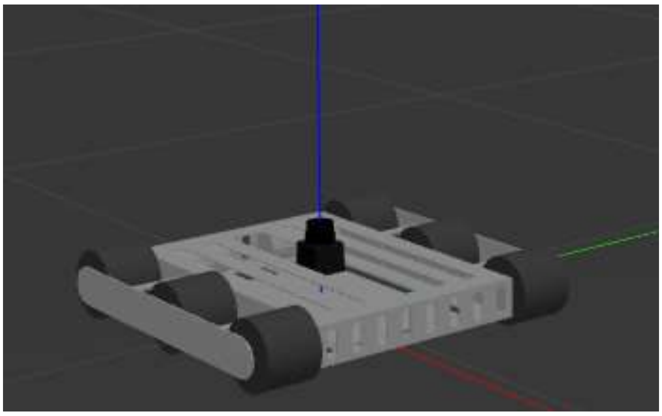


Figure 15:URDF displaying in gazebo world.

6. Mapping and Navigation

6.1 Slam-gmapping node

Using slam-gmapping, you can create a 2-D occupancy grid map (like a building floorplan) from the laser and pose data collected by a mobile robot [8]. Laser data is obtained from laser scans produced by LIDAR (Light Detection and Ranging), pose data is gained through base-link position, and according to these data the robot can calculate the nearest obstacle position regarding the robot’s position, and it draws a map with this given information, As shown in figure (16).

DOI: 10.31695/IJASRE.2021.34020

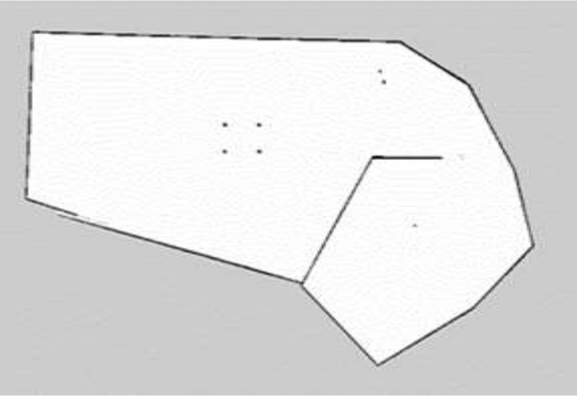


Figure 16:saved map using the laser scans.

6.2 AMCL node

Amcl is a probabilistic localization system for a robot moving in 2D. It implements the adaptive Monte Carlo localization approach, which uses a particle filter to track the pose of a robot against a known map [8,12]. According to Monte Carlo's localization principle, the robot assumes many positions with various orientations these samples are filtered to find the correct position with the correct orientation so that the robot can identify its right position, As shown in figure(17).

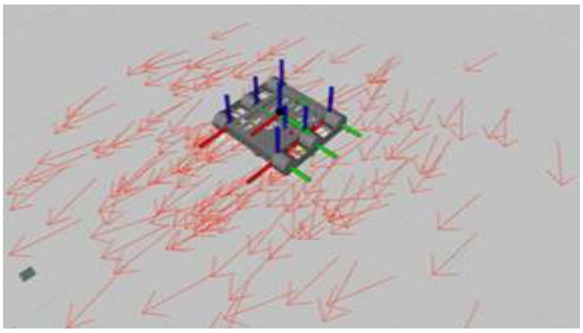


Figure 17:possible assumptions for robot position and

6.3 Move_base node.

given a goal in the world, will attempt to reach it with a mobile base. The move-base node links together a global and local planner to accomplish its global navigation task [8]. Where the global planner is the algorithm responsible for making the original path to reach its goal according to the global cost map, while the local planner is responsible for rechecking if this path is still safe if not it redirects the robot to a new safer path furthermore it operates in a local cost map.

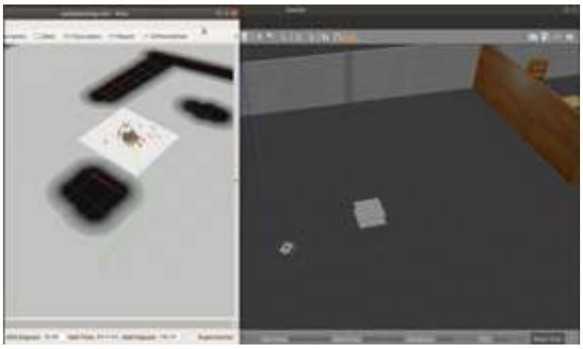


Figure 18:path planned by the robot to reach its goal

7. IMAGE PROCESSING

Object recognition is a multi-task computer vision process, to achieve the required objective it is divided into four tasks, by inputting an image or video feed we utilize: Image Classification, Object Localization, Object Detection, and Object Segmentation [9].

A Model is required to perform all four tasks to achieve a detected object in an image. YOLOv3 was most appropriate to the moving robot application.

7.1 Theory of operation

7.1.1 Layering Process

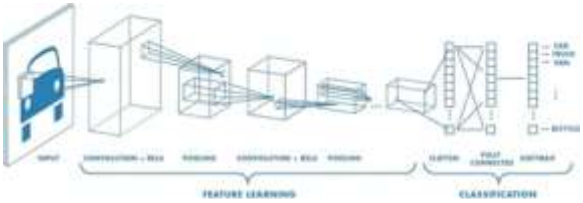


Figure 19:layering process diagram

7.1.2 Convolutional Layer

The job of the convolutional layer is feature extraction. It learns to find spatial features in an input image. This layer is produced by applying a series of image filters to an input image. A filter is a small grid of values that slides over the input image pixel by pixel producing a filtered output image that will be of the same size as the input image.

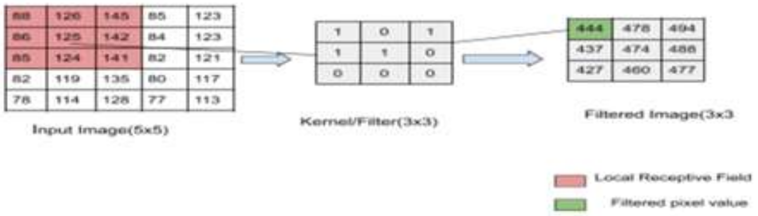


Figure 20:convolutional layer matrix

The **Stride** is how many pixels the filter moves along the image. So, for a value of 1 stride, the filter moves one pixel per process.

7.1.3 Pooling layer

After the convolutional layer comes the pooling layer; the most common type of pooling layer is max-pooling layer. The main goal of the pooling layer is dimensionality reduction. A max pooling operation breaks an image into smaller patches. A max pooling layer is defined by a patch size and stride.

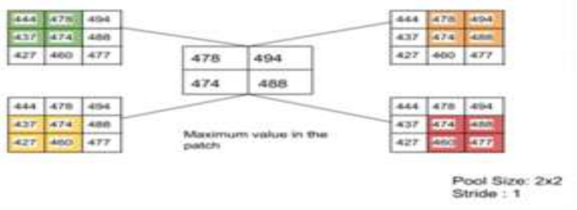


Figure 21:pooling layer matrix

The pooling layer is built by using the maximum value in a patch, depending on the stride of the configuration.

7.1.4 Fully Connected Layer

The last layer in CNN is the fully connected layer. Every output that is produced at the end of the last pooling layer is input to each node in this fully connected layer. The role of the layer is to produce a list of class scores and perform classification based on image features that have been extracted by the layers.

DOI: 10.31695/IJASRE.2021.34020

7.2 The Final Result

By building the matrix as shown in figure(22), for each boundary box **YOLOv3** model is able to achieve the 3 tasks related to object detection and outputting on a user-friendly screen. Comparing pixel data from the object to the data-set provided, a Class of the object in the image is given. CAR class is given for the example above.

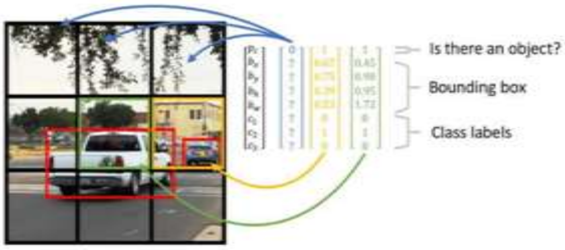


Figure 22:final result

7.3 Test Results

Based on a 2 MP digital USB camera, we were able to obtain satisfactory results under poor lighting, an indication of the feasibility of the YOLOv3 model.

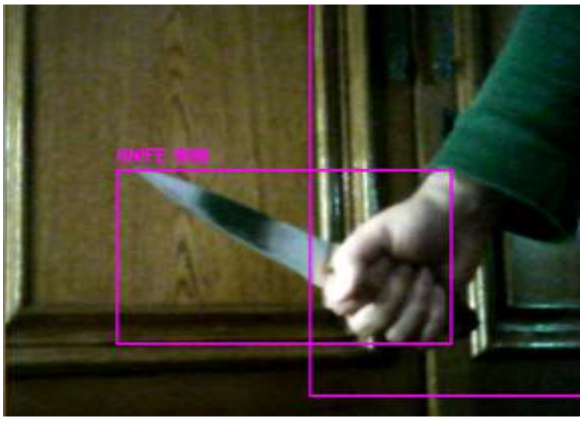


Figure 23:objects, one full object, and one partial object:

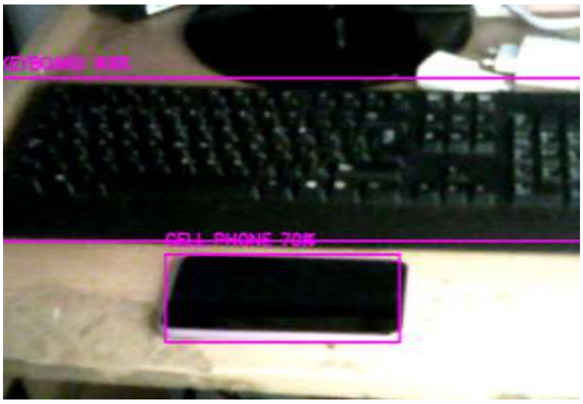


Figure 24:2 different objects, with similar color and lightning but different geometry.

DOI: 10.31695/IJASRE.2021.34020

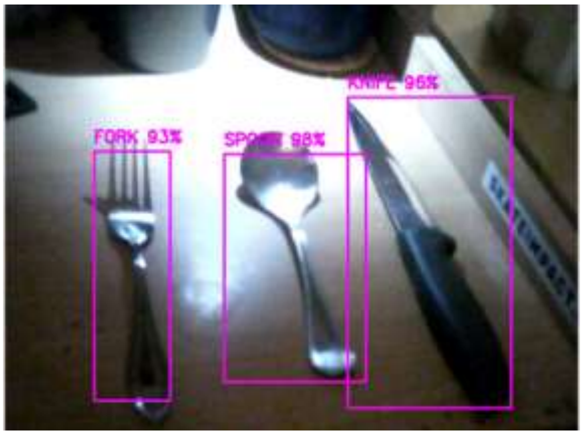


Figure 25:3 objects with similar overall geometry but different in details

8. CONCLUSION

An autonomous navigation system of a tracked robot has been described in this paper. First, Tracked Robot Structure and Design was presented, and finite element analysis was summarized. It was followed by highlighting the dynamic and Kinematic model for the tracked robot. Then according to equations of motion, we build the block diagram of the robot dynamics. And it was followed by a Description of Hardware and ROS-Integration for mapping and navigation any environment. At last, object recognition for many applications is provided by using image processing. The main advantage of the presented system is its capability of adaptation to road segments of different curvatures and the transitions between them. Finally, the obtained simulation and experimental results show the efficiency, of the proposed tracked robot. Future works lie on implementing under ROS more complicated and sophisticated controller methods as well as improving the vehicle's self-localization.

REFERENCES

[1] Ahmad Abu Hatab, R. D. (2013). Dynamic modeling of differential-drive mobile robots using Lagrange and Newton-Euler methodologies: A unified framework. Advances in Robotics & Automation, 02(02).

[2] Bekker, M. G., & Bekker, M. G. (1962). Mechanics of off-the-Road locomotion. Proceedings of the Institution of Mechanical Engineers: Automobile Division, 16(1), 25-44.

[3] (n.d.). Caliber Design - Project-Based Engineering Contractors.

[4] Desoky, A., Bayoumy, A., & Hassaan, G. (2017). Implementation of vision-based trajectory control for autonomous vehicles. International Conference on Aerospace Sciences and Aviation Technology, 17(AEROSPACE SCIENCES), 1-18.

[5] Desoky, A., Bayoumy, A., & Hassaan, G. (2017). Modeling and simulation of a vision-based autonomous vehicle. International Conference on Aerospace Sciences and Aviation Technology, 17(AEROSPACE SCIENCES), 1-23.

[6] Dube, R., Gawel, A., Cadena, C., Siegwart, R., Freda, L., & Gianni, M. (2016). 3D localization, mapping and path planning for search and rescue operations. 2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR).

[7] Dudek, G., & Jenkin, M. (2010). Computational principles of mobile robotics. Cambridge University Press.

[8] Fernández, E., Crespo, L. S., Mahtani, A., & Martinez, A. (2015). Learning ROS for robotics programming. Packt Publishing.

[9] Joseph, L. (2015). Learning robotics using Python. Packt Publishing.

[10] Ko, M. H., Ryuh, B., Kim, K. C., Suprem, A., & Mahalik, N. P. (2015). Autonomous greenhouse mobile robot driving strategies from system integration perspective: Review and application. IEEE/ASME Transactions on Mechatronics, 20(4), 1705-1716.

[11] Marder-Eppstein, E., Berger, E., Foote, T., Gerkey, B., & Konolige, K. (2010). The office marathon: Robust navigation in an indoor office environment. 2010 IEEE International Conference on Robotics and Automation.

DOI: 10.31695/IJASRE.2021.34020

[12] Menegatti, E., Michael, N., Berns, K., & Yamaguchi, H. (2015). undefined.

13] Quigley, M., Gerkey, B., & Smart, W. D. (2015). Programming robots with ROS: A practical introduction to the robot operating system. O'Reilly Media.

[14] Rintanen, K., Mäkelä, H., Koskinen, K., Puputti, J., Sampo, M., & Ojala, M. (1995). Development of an autonomous navigation system for an outdoor vehicle. Intelligent Autonomous Vehicles 1995,

[15] Sebastian, B., & Ben-Tzvi, P. (2018). Physics based path planning for autonomous tracked vehicle in challenging terrain. Journal of Intelligent & Robotic Systems, 95(2),

[16] Vasiljević, G., Miklić, D., Draganjac, I., Kovačić, Z., & Lista, P. (2016). High-accuracy vehicle localization for autonomous warehousing. Robotics and Computer-Integrated Manufacturing,

[17] Wong, J. Y. (2001). Theory of ground vehicles. John Wiley & Sons.

[18] Xing, W., Peihuang, L., Jun, Y., Xiaoming, Q., & Dunbing, T. (2014). Intersection recognition and guide-path selection for a vision-based AGV in a bidirectional flow network. International Journal of Advanced Robotic Systems,

DOI: 10.31695/IJASRE.2021.34020