WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Modelling of Path Control System of Unmanned Ground Vehicle (UGV) to Maneuver Obstruction

[1] Okechukwu Stanley Ikwunze; Innocent Ifeanyi Eneh [2]

[1] Department of Electrical/Electronic Engineering, Abia State Polytechnic, Aba Abia State Nigeria.
[2] Department of Electrical/Electronic Engineering, Enugu State University of Science and Technology, Enugu State Nigeria.
[1]OK_stanlo@yahoo.com, [2]innocent.ifeanyichukwu@esut.edu.ng

ABSTRACT

The tools used in the simulation processes for effective achievement of the goals are Copernicus, Gazebo Simulator, Solidworks and Rviz (ROS visualization). The Copernicus was used because of its packages with predefined maps and configuration files to help the robot navigate and Solidworks was used because of its ability to apply the parametric design to generate the part, the assembly and the drawing of the rover robot. The Gazebo Simulator was used as a 3D simulator with the ability to accurately and efficiently simulate populations of robot in complex indoor and outdoor environment to have the behaviour of a robot in real world (physical environment). Then the Rviz which is 3D software visualization was used to enable visualization of state of the robot using sensor data. The controller that was employed in the modelling was particularly the "Remote-Controlled Obstacle Avoidance" which comprised of three controlling system (remote control and obstacle avoidance). These amalgamated or integrated control systems were used to arrive at the point of interest of this work which includes avoidance of obstacle and control to be capable of work effectively in dangerous environment. The result of the modelled path control system of the Unmanned Ground Vehicle (UGV) shows that the UGV was able to meander through the environment (software environment) from one unit or section of the environment to another without a hitch (obstruction) and was able to map the terrain of the environment. It identified the obstructions and maneuvre them.

Keywords – Path Planning, Control, Development, UGV, Maneuver, Obstruction

INTRODUCTION

Unmanned ground vehicle is an artificial intelligence (AI) whose activities are to perform tasks better, faster and more accurately and efficiently than humans, especially when it concerns repetitive and detail-oriented task. Artificial Intelligence involves using methods based on the intelligent behaviour of humans and other animals to solve complex problems [1]

It is important to model the UGV path tracking control system. The path consists of ordered waypoints in the areas (planes) which the UGV (rover) is desired to pass through [2]. These waypoints (terrains) are assumed to be provided by a high-level path planner and would represent an obstacle free path for the rover. These waypoints can be loaded using the roverDesiredPath.mat file [3].

In control, the controllers receive low-level real-time vision features as input only, which is a deep knowledge from a replication Kinect sensor, and control the hand of robot through explicit specification of the desired grasp orientation and location. Supporting this mannerism, numerous past studies had displayed evolutionary approaches as promised in enabling moving behaviour generated in autonomously oriented robots [4, 5]. Additionally, speeding up the pattern recognition processes in which concept of reality parallelizes the learning replication by multi-threading the NEAT algorithm. A search in evolutionary demands a suitable function that ascertains the attribute of person's remedies [6, 7]. Due to the fact that the grasping mannerism robustness is desirable in such experiment, therefore it is important to consider measuring of the quality of grasping technique. Former gripping technique researches focused concisely on the type of contact, positions, hand geometry and kinematics [8].

For an UGV to move freely in any environment, it should possess the feature of maneuvering obstructions in the paths of the terrain. This brings about the goal of this work, which is to model a path tracker system that will enable the UGV move hitch-free along the paths of the terrains in the environment.

MATERIALS

The materials used include the software and hardware. The Software used in this work for the modelling and simulation processes of the rover (UGV) was GAZEBO Simulator. To achieve real-time communication, additional software requirements were employed and such software includes: Microsoft

Windows Software Development Kit (SDK) 7 Compiler and .NET Framework 4. The Windows 7 SDK provides the latest headers, libraries, metadata, and tools for building Windows 7 applications (Microsoft.com, 2017). The Windows 7 SDK, when used in conjunction with Visual Studio 2010, provides the best experience for instigating various applications. The hardware include laptop computer is an Intel core i7 CPU running on Windows 8 operating system; Raspberry Pi which is use as the central controller collecting all data from the input sensors and driving the robot according to the coordination that we are given; input devices (Lidar Module, GPS Module, Web/Depth Camera, etc.); output devices (Gear Motors, Motor Drivers, Batteries, etc.)

UGV (ROVER) PATH PLANNING AND CONTROL

This section describes the modelling of rover's (UGV) path tracking control system. The path consists of ordered waypoints in the areas (planes) which the rover is desired to pass through. These waypoints (terrains) are assumed to be provided by a high-level path planner and would represent an obstacle free path for the rover. These waypoints can be loaded using the roverDesiredPath.mat file.

The goal of this subsystem is to first, compute the necessary steering angles and the wheel speeds needed to follow a desired path and a desired chassis linear velocity and second, to compute the necessary actuator torques needed to achieve these steering angles and wheel speeds [9].

For developing the path tracking controller, the following considerations were made:

- The rover is typically assumed to have low forward velocity (on the order of cm/s); therefore, the dynamics of the motion are ignored and the controls problem is approached using kinematic equations only.
- To simplify the kinematics formulation, the rover is assumed to be moving on a 3-dimensional surface.
- The six wheels of the rover have independent steering which can enable the rover to perform Ackerman steers. Based on this capability, the rover is considered to be using Ackerman steering.
- The Ackerman steering geometry is simplified by assuming a 3D geometric vehicle model with an equivalent turn radius. This simplification is done by representing each pair of wheels by a single wheel located in the middle and a single steering angle corresponding to the turn radius of the center of the rover (UGV).
- The front, middle and the rear wheels are considered to be steered symmetrically.
- The wheels are assumed to roll without slipping.

Based on the above considerations, the six-wheel rover can be equivalently represented by a geometric bicycle model.

Robotics System Toolbox was used for path tracking. This is a geometric algorithm that computes a target direction angle (α) needed to move the robot from its current position to reach some look-ahead point in front of the robot.

MODELLING OF THE PATH CONTROL SYSTEM OF THE UGV

The modelling has two sections which are the launching of a Gazebo Simulation Environment and Mapping an Environment with Copernicus.

A Gazebo simulation is a robot simulation made with Gazebo, a 3D simulator with the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments. Copernicus is a ground outdoor mobile robot platform that is used in researches involving robotics. It was used to support the Robot Operating System (ROS) out of the box and allowed entrance to motion commands through a sequential boundary. This platform is designed to support multiple third part sensors and manipulators and the support covers the software, together with, the electromechanical integration of the sensors needed in such a way that the effort put in to set up the requiring platform was negligibly small. The version of ROS used is 1.1.0 packages which were used to run the Copernicus with hardware.

Launching a Gazebo Simulation Environment

Open the terminal and enter the following commands one by one:

- echo "source /usr/share/gazebo-9/setup.sh" >> ~/.bashrc
- echo "export GAZEBO_MODEL_PATH=~/copernicus_ws/src:$GAZEBO_MODEL_PATH" >> ~/.bashrc
- echo "export GAZEBO_PLUGIN_PATH==$GAZEBO_PLUGIN_PATH" >> ~/.bashrc
- source ~/.bashrc
- roslaunchcopernicus_simulationgazebo.launchwillow_garage:=true
- roslaunchcopernicus_simulationsimulation.launch
- rosrunrvizrviz

Open a new terminal for the next command. In the new terminal, enter:

- roslaunchcopernicus_teleoperatorteleoperator.launch keyboard:=true
- This will launch the keyboard teleoperator node
- The keyboard controls for the teleoperator are as follows:
- To move around:

U i o

J k l

M , .

- anything else: stop
- q/z : increase/decrease max speeds by 10%
- w/x : increase/decrease only linear speed by 10%
- e/c : increase/decrease only angular speed by 10%

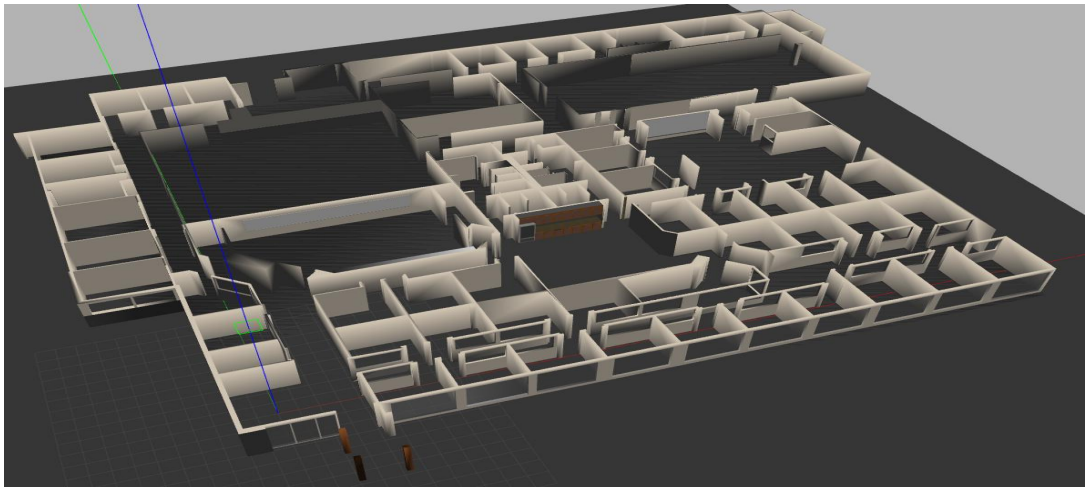

Fig. 1: Image of the environment and its terrains

Mapping an Environment with Copernicus

Launch the move_base node by running:

$ roslaunchcopernicus_basebringup.launch

Launch the sensors node by running:

$ roslaunchcopernicus_basesensors.launch

To perform the mapping, launch the navigation package by running

$ roslaunchcopernicus_navigationnavigation.launchgmapping:=true

Launch the rviz visualization tool by running:

$ rosrunrvizrviz

You can then open the copernicus configured rviz environment by opening the copernicusrviz config file, located under copernicus_navigation->rviz_config->navigation.rviz, from the rviz tool

In order to control the robot, launch the teleoperation node by running:

$ roslaunchcopernicus_teleoperatorteleoperator.launch keyboard:=true

Once the mapping of the entire environment is completed, the map can be saved by running:

$ rosrunmap_servermap_saver –f <filename>

You can then open the copernicus configured rviz environment by opening the copernicusrviz config file, located under copernicus_navigation->rviz_config->navigation.rviz, from the rviz tool.

Use the 2D Nav Goal tool in the top toolbar to select a navigation goal in the visualizer. Ensure that the nav goal is given in a mapped section of the map.

In a new terminal, enter:

roslaunchcopernicus_navigationnavigation.launchgmapping:=true

On a separate terminal, launch the rviz visualization tool by running:

rosrunrvizrviz

- Once the mapping of the entire environment is completed, the map can be saved by running:
- cd ~/copernicus_ws/src/copernicus/copernicus_navigation/maps
- rosrunmap_servermap_saver -f <filename>

Once the new environment has been mapped, move the map file to the "maps" folder located under copernicus_navigation package

Update the map filename in the navigation.launch file:

In navigation.launch, locate this line: <arg name="map_file" default="$(find copernicus_navigation)/maps/<map_filename>"/>

Replace <map_filename> with the name of the newly saved map file.

In a new terminal, enter:

- roslaunchcopernicus_basesensors.launch
- Use the 2D Nav Goal tool in the top toolbar to select a navigation goal in the visualizer. Ensure that the nav goal is given in a mapped section of the map

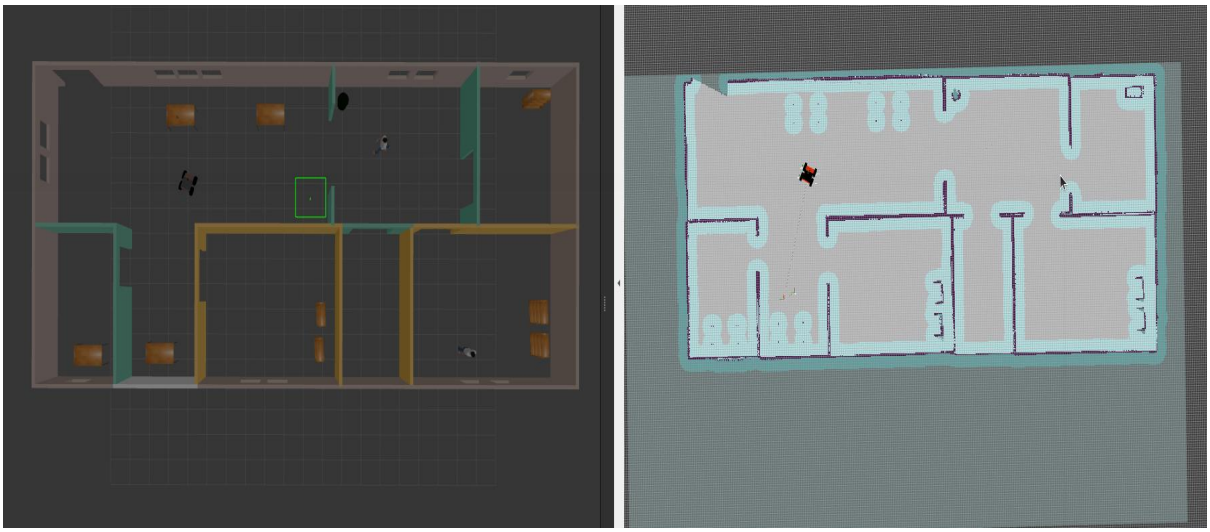

Fig. 2: Image of the environment and the generated map

CONCLUSION

With the use of tools like Copernicus, Gazebo Simulator, Solidworks and Rviz (ROS visualization) in the simulation processes the path tracker system of the UGV was modelled successful and the UGV was observed to move through the software environment without coming in contact with any obstruction. The software environment is separated by units or sections with connecting paths. The UGV movement in the terrain was very free and smooth moving in and out of the different units with no difficulties (obstructions). The obstacles like doors, walls, etc., were identified by the UGV and it avoided them by taken the correct path when it was mapping the environment.

RECOMMENDATION

The UGV can be employed in distribution of files from one office to another in organizations since it can move freely without glitches in a given arena (environment).

CONFLICT OF INTEREST

No conflict of interest between the authors.

REFERENCES

[1] Javier, A. P., Fani, D., Daniele, R., Guang-Zhong, Y. (2016). Artificial Intelligence and Robotics. UK-RAS Network Robotics & Autonomous System. P. 56.

[2] Bongard, J. C. (2008). Behavior chaining: incremental behavioral integration for evolutionary robotics. In Artificial Life XI. Cambridge, Massachusetts: MIT Press, pp. 64–71.

[3] Bongard, J. C. (2010). The utility of evolving simulated robot morphology increases with task complexity for object manipulation. In *Artificial Life*, vol. 16(3), pp. 201–223.

[4] Miller, A. T., Allen, P. K. (1999). Examples of 3d grasp quality computations. *IEEE International Conference on Robotics and Automation*, vol. 8, pp. 1240–1246.

[5] Mita, T., Yamaguchi, T., Kashiwase, T., Kawase, T. (1994). Realization of a high speed biped using modern control theory. Int. J. Control, vol.40 (1). Pp. 107-119.

[6] Sala, A., Bruci, M. (2017). Proposed robot scheme with 5 DOF and dynamic modeling using maple software. J. of mechanical engineering. Vol.67(2), pp. 101-108.

[7] Shala, A. andBruqi, M. (2017). Trajectory Tracking of Mobile Robot using Designed Optimal Controller. *International Journal of Mechanical Engineering and Technology,* vol.8(8), 649 – 658.

[8] Siddique, N. and Adeli, H. (2013). “Computational Intelligence Synergies of Fuzzy Logic, Neural Networks and Evolutionary Computing,” West Sussex: John Wiley & Sons, Ltd.

[9] Filip Jan; Martin Azkarate and Gianfranco Visentin (2017). "Trajectory control for autonomous planetary rovers." In *14th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA)*.