Journal of Mechatronics and Robotics

Original Research Paper

LiDAR-Based Obstacle Detection and Avoidance for Navigation and Control of an Unmanned Ground Robot Using Model Predictive Control

Sai Charan Dekkata, Sun Yi, M. A. Muktadir and Selorm Garfo

Department of Mechanical Engineering, North Carolina A&T State University, Greensboro, United States

Article history
Received: 02-02-2023
Revised: 04-02-2023
Accepted: 22-02-2023

Corresponding Author:
Sai Charan Dekkata
Department of Mechanical Engineering, North Carolina A&T State University, Greensboro, United States
Email: sdekkata@aggies.ncat.edu

Abstract: Unmanned Ground Vehicles (UGVs) have, as of late, been utilized in a wide assortment of utilizations because of their flexibility, diminished expense, and quick response, among other benefits. Search and Rescue (SAR) is quite possibly the most conspicuous zones for the work of UGVs instead of a monitored mission, mainly due to its impediments on the expenses, human resources, and view of the human administrators. An ongoing way of arranging to utilize numerous helpful UGVs for the SAR mission is proposed in this study. This study aims to introduce the initial moves towards a Model Predictive Control (MPC) based peril evasion calculation for UGVs representing the vehicle elements through high constancy models and uses just surrounding data about the environment as given by the available onboard sensors. In particular, the paper presents the MPC definition for peril evasion utilizing a Light Detection and Ranging (LiDAR) sensor and applies it to a contextual of the effect of model constancy on the calculation's presentation, where execution is estimated principally when to arrive at the objective point. The Robot Operating System (ROS) is used to drive the sensors and visualize the data in RVIZ. This study presents MPC development for navigating Husky A200 by adjusting the longitudinal, lateral, and yaw motion command behaviors. The proposed algorithm for Husky A200 is tested indoors and compared the results with the simulation results plotted using MATLAB and GAZEBO. A novel simulator package is developed for the Husky using RVIZ and GAZEBO. The efficiency of the proposed MPC design is tested through simulation and compared with real world experiments, the real-time longitudinal movement follows the simulation results closely. For MPC's short-term optimization, an optimized control signal from a linear framework is utilized for a linear quadratic controller. According to the Husky position and orientation, applying a transformation to convert the map coordinate system to the Husky coordinate system. Transforming the map coordinate system helped in computing the errors because the initial vector considers position and orientation as zero.

Keywords: Model Predictive Control, Unmanned Ground Robot, Autonomous Vehicles, Robust Adaptive Control, Vehicle Dynamics, Control and LiDAR

Introduction

Model Predictive Control (MPC) has been of great interest to both industry and academia when initial versions of the MPC are for Dynamic Matrix Control (DMC) and Identification Command Algorithm (ICA). The terminal constraints, terminal cost function, and local control law are the three essential ingredients that guarantee the MPC's stability (Mayne and Michalska, 1988). Despite MPC's significant practical importance and general use, almost no hypothesis to direct these regulators design and tuning for execution, particularly in the non-linear system, is complicated and rarely implemented. MPC generates discontinuous control

moves for implementing simplified computations for single input single output and multi-input and multi output process models. The MPC is robust for many reasons and forms a vital problem class for many applications and arises as subproblems, in general, constrained optimization methods, such as sequential programming or interior point methods (Goodwin, 2005). Having the waypoints generated by the planner and the GPS coordinates, MPC can consider future waypoints/events and act accordingly. Convex optimization is a subclass of numerical streamlining with favorable hypothetical and functional properties, which will be used as significant control parameters (Kirches, 2011). To some extent, step response models are a subset of the transfer function. MPC encompasses a range of control methods used in the single input, single output, multi-input, and multi output processes (Gardezi and Hasan, 2018). In this research, the initial step is to recurrent some fundamental theoretical results and outline the optimization method used for Husky A200. The traditional MPC issue is settled by executing an improvement routine, which has limited the use of model regulators to moderate dynamic measures. Lately, various strategies have been created with the objective of optimizing MPC to be utilized for quickly inspected frameworks.

Over a considered prediction horizon, find the $J(u)$ and use the non-linear optimization technique (Muraleedharan *et al*., 2021). Where Q is the positive semi definite, R is the positive definite. As mentioned, the sequence of control inputs U, a series of desired states X_d, and the sequence of predicted states X would minimize the unknown disturbances and mode the Husky dynamics by representing the Gaussian process. With favorable practical and theoretical properties, mathematical optimization has a convex optimization subclass (Boyd *et al*., 2004). The speculation of the classical function hypothesis can be productively utilized in multiple mathematical analyses. Here articulating the theoretical implicit function, which uses the stability analysis for the control algorithm improves the sequence of predicted states. Because of the progression in computational forces, practical calculations are being utilized. The model infers a mathematical connection between network weights and the exchange transfer function limits. For multi-input multi-output systems in the model, a distinguish technique algorithm was proposed using the closed loop method.

Husky A200

The Husky series has been used by industrial and military engineers and robotic researchers who spend time researching and developing UGV prototypes. The Husky A200 Unmanned Ground Vehicle (UGV) is engineered to thrive in harsh outdoor and indoor conditions. Husky has rugged all terrain tires, a powerful 4 × 4 zero maintenance drivetrain, class leading ground clearance, and ample internal space for custom hardware. For extreme terrain, 13 lug tires with a high torque 4 × 4 drive are available. On top of the Husky industrial standard rail for rapid payload, mounting is available. Charging is made easy with a field chargeable 24 V battery with a toolless access panel. Husky bundle involves fundamental segments, as it comes pre-introduced with a small ITX with Linux and ROS. The top plate considers the simple mounting of any detecting, control, or PC equipment associate sensors to the locally available PC and Husky managed power supplies to begin Husky A200.

Simulation

Initially, for obstacle detection using LiDAR, simulation results are plotted using MATLAB and GAZEBO, using the results are compared. Robot Operating System (ROS) is used to drive the sensors, launch packages containing the sensor fusion algorithms, and visualize the data in RVIZ. A novel simulator package was designed by modeling the Husky using RVIZ and GAZEBO. GAZEBO's powerful physics engine and cross compatibility with ROS allowed for an easy way to test essential torque and angle inputs. A customized scene can be generated with simulated vehicles in the background using the math works automated driving toolbox. Because of ROS's heavy use for the actual implementation, Gazebo made it easy to move back and forth from the simulated environment to accurate testing of the vehicle with a slight modification to the software. This study presents MPC development instead of a conventional PID for navigating the Husky A200 by adjusting the longitudinal, lateral, and yaw motion command behaviors. The purpose of using MPC instead of PID is its ability to optimize current time slots while taking future time slots into account to provide a more proactive and smoother control mechanism. The planner and the GPS coordinates generate the waypoints, and MPC considers future waypoints/events and acts accordingly.

Simulation

MATLAB and Simulink

The LiDAR sensor data has been recorded using a 3D simulation environment to test the algorithm. The recorded data helps to develop the perception algorithm. Simulating sensor detections and testing the reliability and predicting potential complex events during autonomous driving can be generated in the simulation environment (Muktadir and Yi, 2021). In the automated driving scenario generation, vehicle models are defined and help to create a road network and control the simulated vehicle behavior using the driving scenario application.

In the simulation environment, multiple scenes are available in the automated driving toolbox. To create a driving scenario and place a moving vehicle in the background, LiDAR has been configured and placed on the vehicle using MATLAB. The perception algorithm is tested

DOI: 10.3844/jmrsp.2023.27.41

in the simulation environment with different static and dynamic objects. The developed algorithm can be used in the simulation environment for a wide range of custom scenes and tested on numerous available scenes in the toolbox.

Figure 1, ROS has been used to view the LiDAR sensor point cloud and LiDAR sensor properties are configured. The automated driving toolbox has been used to integrate a 3D simulation environment to view sensor readings. LiDAR is mounted on top of the vehicle's front center and configured to model a typical Velodyne VLP-16. The point cloud data is recorded through the simulated sensor over the vehicle and data is visualized. The data is filtered for any noise or disturbance before testing the algorithm. A 3D map is built in the simulation environment using the LiDAR sensor data and the perception algorithm is tested in the simulated environment. If multiple moving objects are defined in the simulation, the algorithm is tested for extracting the moving vehicle's distance from the LiDAR point cloud.

When the record mode is set to true in the scenario setup, the model records and visualizes the synthetic LiDAR data; when the record mode is set to false, the model runs the perception algorithm. During the simulation, the record mode is closed, and visualize the subsystem and develop a way to record data. Updating the simulation stops time to end when the reference path is completed, then the simulation can be executed. Finally, from the recorded data, a point cloud data array is created. A LiDAR map builder is designed and loops through the point cloud array and updates the map with a new LiDAR frame and updates the top view display. To build the LiDAR map, closing the high-definition figure recorder and setting the record mode and algorithm can be stress tested under different scenarios.

The Simulink model is modified to generate code for path planning and vehicle control algorithms and verify the generated code using Software in the Loop (SiL) simulation. In the implementation and simulation, it helps to generate and verify code from the algorithm model. The algorithm test bench is used to partition the algorithm and test bench specifying the path planning and vehicle control functionality. The test bench model defines the stimulus and environment to test the algorithms. Simulating the test bench to test the algorithm model reference in various simulation models and the cost map creator block creates the environment's cost map. The behavior planner block triggers navigation tasks and the controller is applied to the vehicle model block. The path planner plans a feasible path through the environment map and the trajectory generator smoothens the reference path by fitting splines. The vehicle controller controls longitudinal and lateral movement. The variable size poses signal has been split into a fixed size outport and the cost map bus is associated with the cost map input port.

Configuring the model to generate code includes setting C++ code with entry points for each rate and applying standard optimizations. A report is generated to facilitate exploring the generated code and finally, set and view model parameters to enable C++ code generation. SIL helps test the source code on our development computer. The numerical test equivalence between the model components and production code generated from the elements by using Software in the Loop (SiL) and Processor in the Loop (PiL) simulations.

Figure 2, shows the architecture of the unreal engine connected to the bridge model on a separate MATLAB instance which transfers messages like speed, vehicle controls, and sensor readings. The model is a proportional controller introduced in the system to configure a model to generate C++ code for standalone ROS nodes. Configuration of the connection to the ROS device in the Simulink coder, the code generated for the standalone ROS node is connected to the ROS device. The standalone ROS node is generated and can automatically transfer, build, and run on the ROS device. Tested the generated ROS node using a ROS master running on the ROS device. Finally, the newly built ROS node is executed, and its behavior is verified using a MATLAB-based robot simulator. ROS nodes generated can be managed by simulink with the ROS device object and stop the node at any point without rebuilding it in the simulink.

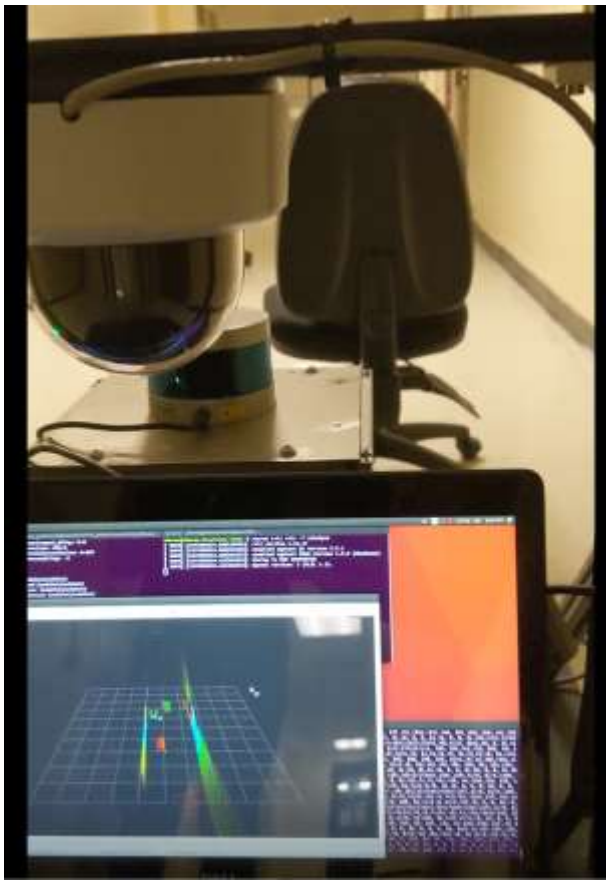

Fig. 1: LiDAR 3D point cloud

DOI: 10.3844/jmrsp.2023.27.41

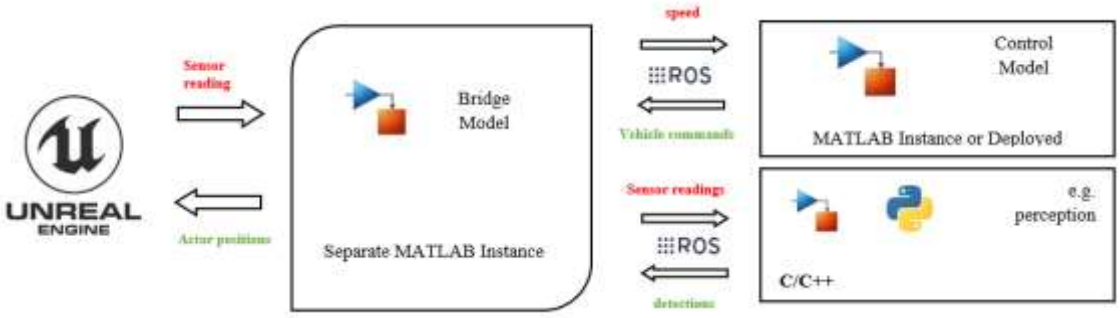

Fig. 2: Unreal engine connected to ROS.

Gazebo and RVIZ

ROS is the middleware opted for due to its incredible support and its broad compatibility with the sensors. ROS is used to drive the sensors, launch packages containing the sensor fusion algorithms, and visualize the data in RVIZ. The LiDAR sends information about the environment in front of the Husky, including static and dynamic objects; this data can be visualized in RVIZ. Husky is modeled using differential drive equations; a novel simulator package was designed by modeling the Husky using RVIZ and Gazebo. Gazebos' powerful physics engine and cross compatibility with ROS allowed for an easy way to test essential torque and angle inputs as well as testing GPS path tracking algorithms (Vougioukas *et al*., 2007).

Rigid body dynamic equations are:

$$\dot{x}(t) = v(t)\cos\theta(t) \tag{1}$$

$$\dot{y}(t) = v(t)\sin\theta(t) \tag{2}$$

$$\dot{\theta}(t) = \omega(t) \tag{3}$$

In this research, an integral reinforcement learning method is used to decide the ideal control framework with totally obscure elements. The proposed approach assesses the ideal control utilizing essential support, using the framework distinguishing proof rather than function approximation. This methodology delivers the test process to control with the automated ground vehicle elements ideally (Gao *et al*., 2010):

$$P(z): Minimize_{u_0} \sum_{i=1}^{N-1} \left\| x_t - x^{ref} \right\|_Q^2 + \left\| u_t \right\|_Q^2 + \left\| x_N - x^{ref} \right\|_{QN}^2 \tag{4}$$

Subject to:

$$x_{t+1} = f\left(x_t, u_t\right), t = 0,...., N-1 \tag{5}$$

$$u_{\min} \le u_t \le u_{\max}, t = 0,...., N-1 \tag{6}$$

$$x_0 = z \tag{7}$$

where, x_0 is the position and orientation of the Husky. $u = (v, w)$ is the linear and angular velocity, and x^{ref} is the orientation and target position:

$$x = \left(p_x, p_y, \theta\right) \tag{8}$$

$$f(x,u) = \begin{bmatrix} p_x + t_s\left(v\cos\theta\right) \\ p_y + t_s\left(v\sin\theta\right) \\ \theta + t_s w \end{bmatrix} \tag{9}$$

Gazebo is a 3D simulator proposed to have rational recreations of almost realistic test scenarios of mechanical situations. Simultaneously, ROS fills in as the Husky interface, consolidating bring about an incredible robot test system. With Gazebo, a 3D problem can be recreated on the computer with obstacles and numerous different and complex situations. The Gazebo likewise utilizes a physical engine for illumination, inertia, and gravity. Advantages of the Gazebo help in assessing and testing the Husky in troublesome and hazardous conditions with no harm to the robot. It is regularly quicker to run a test system as opposed to starting the Husky. To create a simulation, launching the files in the Gazebo gives a 3D model of the Husky and a portrayal of joints and sensors. The ROS interface is designed to suit Husky's joints to establish a workspace environment.

The more exact model of the Husky is available in the robot model's directory from a differential drive. Gazebo allows custom worlds and environments to be designed and includes an extensive library of objects, structures, and buildings. ROS's heavy use for actual accurate testing of the Husky with a slight modification to the software. An interesting thing about the designed simulator is that the exact code used for simulation can be implemented for the Husky with almost no change in the code. With the collection of robots and objects, recreating the world for testing Husky under different scenarios is feasible. A few

DOI: 10.3844/jmrsp.2023.27.41

Gazebo objects replicate real world dimensions such as a dumpster, construction cone, fire hydrant, and construction barrel. A few objects in the Gazebo world that have only collision geometry are identified as static objects. Objects which have both collision and inertia geometry are identified as dynamic objects. Simple objects can be added like a box, sphere, or cylinder from the render window. Also, load the model database where each section is labeled with a path.

The pose of each model may be altered through translation and rotation tools. Inserted objects can be moved in the *x*, *y*, and *z*-axis. The three-axis visual maker will appear on the object, which helps move the object in the *x*, *y*, and *z* directions. Rotate tool helps to orient mode around the *x*, *y*, and *z*-axis. Scale tools can resize a model in *x*, *y*, and *z* directions. The generated model is converted to Simulation Description Format (SDF) in the recording registry, as this is a standard way we can work in Gazebo. Nonetheless, utilizing the Unified Robotic Description Format (URDF) document produced by Xarco and moving it in the depiction bundle with ROS. The URDF is an XML record design utilized in ROS as the local arrangement to portray all the Husky components. Xarco is an XML macro language that is valuable to make more limited and more exact robot depictions and the total portrayal needs must be inside the robot tag. Consists of three robot labels; one of the labels is utilized for the visual rendering engine, the second is the physics engine and the last is the collision detection engine. The orange cones in the scene are all static and barriers are placed to act as a wall to block Husky's navigation. All the objects placed in the scene have collision geometry.

RVIZ is a 3D visualization tool for ROS, and various formats of presentations are frequently helpful for multiple uses of the visualizer. A practical design for a full PR2 isn't precious for a test robot. The visualizer allows to load and save of various setups and contains display properties, camera type settings for the initial viewpoint, and tool properties for the underlying perspective. With RVIZ for Groovy, the design document changes from VGC/INI to.rviz/YAML, presumably the inside setup environment. The RVIZ in Groovy isn't in reverse viable, so it is impossible to open or change over old.rviz documents in the Groovys' RVIZ. There are various diverse camera types accessible in the visualizer and the camera types compromise multiple methods of controlling the camera and numerous sorts of projection. The camera essentially turns around a focal point while continuously seeing that point and is pictured as a bit of a circle while moving the camera. RVIZ utilizes the *tf* change framework for exchanging information from the coordinate frame to the global reference frame. There are two facilitate frames out of which the fixed frame is the most important and is also the reference frame used to signify the world frame.

Figure 3 shows, on the left-hand side of the image, a custom-built scenario in Gazebo to test Husky's perception, control, and behavior. On the right-hand side of the picture, LiDAR data is visualized in the 2D format in rviz. The odometry frame on the off chance that the fixed frame is incorrectly set to show the Husky base points towards all the items in the map move concerning the Husky. For better results, the specified frame ought not to be moving comparatively with the global frame.

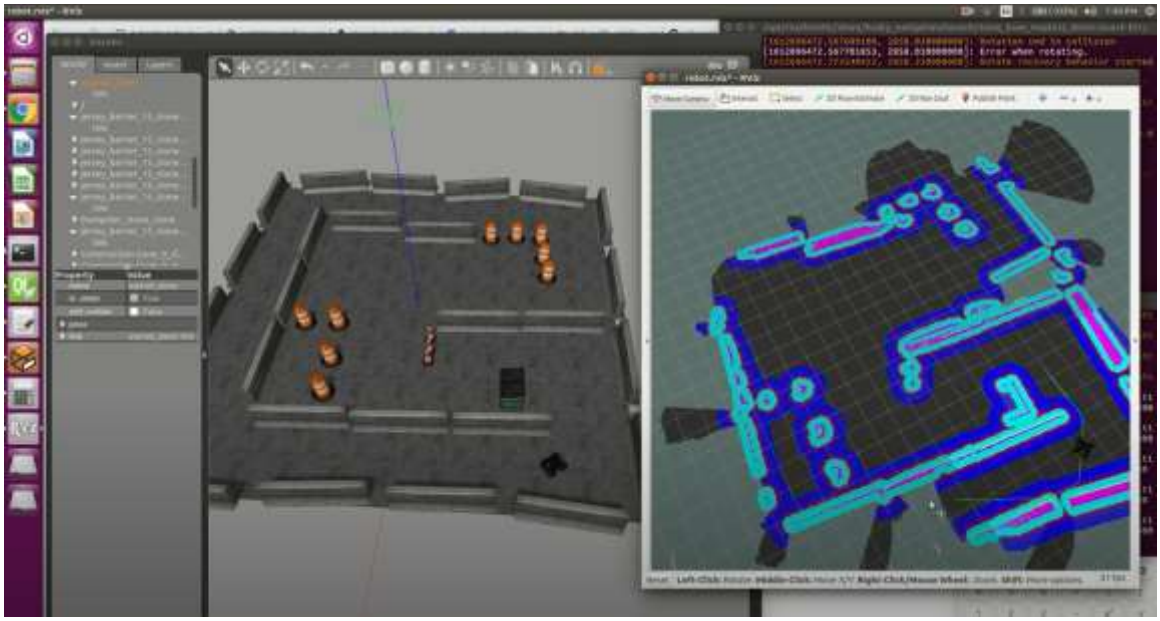

Fig. 3: 2D mapping in RVIZ

The target frame is the reference outline for the camera; if the target frame is the guide, the Husky will be moving around in that direction. If the target frame is the Husky base, the Husky will remain in a similar place while other things move comparatively. The 2D navigation goal lets to set an objective for the ROS points on an area in the ground plane. The 2D pose sets an underlying posture to seed the initial pose for localization to the initial pose ROS topic.

Materials and Methods

Model Predictive Control Design

Despite MPC's significant practical importance and general use, there is almost no hypothesis to direct these regulators' design and tuning for execution, particularly in the non-linear case (Sutton and Bitmead, 2000). The analysis of the MPC stability and robustness to non-linear systems is complicated and rarely implemented. Referring to Eqs. 10-12, which are linear prediction model's general prediction (Mayne *et al*., 2000):

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) + E(t) \tag{10}$$

$$x(0) = x_0 \tag{11}$$

$$y(t) = C(t)x(t) + D(t)u(t) \tag{12}$$

Here we have state time as (t) and manipulated variables are $u(t)$ and $y(t)$ are the controlled variables. The system is called linear time variant if the $A(t)$, $B(t)$, $C(t)$, and $D(t)$ are functions of time (Mayne and Langson, 2001). Equations 13-15 represents a nonlinear system that is time-invariant in most cases and g, and f are considered continuous differential functions (Diehl *et al*., 2002). Based on the goals of this study, objective functions are chosen:

$$\dot{x}(t) = f\left(x(t), u(t), t\right) \tag{13}$$

$$x(0) = x_0 \tag{14}$$

$$y(t) = g\left(x(t), u(t), t\right) \tag{15}$$

Equation 16 assumes; the plant is nth order time invariant multi-input and multi-output process:

$$y(k) = \sum_{i=1}^{n} A(i)y(k-i) + \sum_{i=0}^{n} B(i)u(k-i) + \sum_{i=1}^{n} C(i)v(k-i) + e(k) \tag{16}$$

where, $y(k)$ is a vector of n_y output, $v(k)$ is a vector of n_v disturbances, $u(k)$ is a vector of n_u manipulated inputs, and $e(k)$ is the white noise. MPC generates discontinuous control moves for implementing simplified computations. The model is a set of multi-input single output process models (Dekkata, 2018):

$$y_j(k) = \sum_{i=1}^{n} a_j(i)y(k-i) + \sum_{i=0}^{n} B_j(i)u(k-i) + \sum_{i=1}^{n} C_j(i)v(k-i) + e_j(k) \tag{17}$$

There are two types of constraints: Equality and inequality. But in most of the systems inequality constraints are used (Grüne, 2012). In the model, imposing constraints and limitations on the system's state, and boundary conditions to stay within some bounds and constrain input significantly. So, in many control strategies, the actuation is a continuous variable. Often the controller will make an unrealistic demand if the controller is designed severely. It might command to use of unphysical scenarios, like going a million miles an hour for a fraction of a second and slowing down. MPC can essentially put boundaries on the control signal while running this optimization loop. For every time step when this optimization runs, it can effectively constrain the optimization to hit these maximum or minimum values. So, it can run hard constraints and soft constraints on the state or the input (Wang and Boyd, 2009).

Non-Linear Systems

Linearizing the non-linear model about a particular state and linear optimization are applied to the problem because optimization is the MPC's utmost function. It is built around optimization for every single time step. Consider a couple of approaches but imagine having a linear structure with some noise and disturbances to the bigger picture model. The optimized control signal from a linear framework utilizes a linear quadratic controller for the MPC's short-time optimization. Changing the framework and disturbances' boundaries will repay over time (Richards, 2005).

System Formulation

The system formulation of discrete time control (Zeilinger, 2011):

$$x(k+1) = Ax(k) + Bu(k) + w(k), k \in \mathbb{N} \tag{18}$$

Equation 18 is subject to the below constraints, refer to Eqs. 19-20:

$$x(k) \in X \subset \mathbb{R}^n, \quad \forall k \in \mathbb{N} \tag{19}$$

$$u(k) \in X \subset \mathbb{R}^m, \quad \forall k \in \mathbb{N} \tag{20}$$

Equations 19-20, the control input is $u(k)$, the state vector is $x(k)$, and the disturbance in the system is $w(k)$. Having the

polytopic constraints X and U with two sets of disturbances in the model as convex and compact in the framework of $w(k)$ (Dekkata, 2021). Equation 21 is for the closed loop system with a control law consisting of $u = k(x)$:

$$x(k+1) = Ax(k) + Bk(x(k)) + w(k) \tag{21}$$

In the initial states $\dot{x}(0)$, for an unprecedented system with a closed loop model having disturbances w is formulated using $\emptyset(k, x(0), w)$. Considering an optimal model where having no disturbances in the system is given by Bock and Plitt (1984):

$$\bar{x}(k+1) = A\bar{x}(k) + B\bar{u}(k) \tag{22}$$

Equation 22 for an initial state $\dot{x}(0)$, represented as $\emptyset(k, (0), w)$. The closed loop formulation of the system is given from the below equation, refer to Eq. 23 for a controlled law $u = k(x)$:

$$\bar{x}(k+1) = A\bar{x}(k) + Bk(\bar{x}(k)) \tag{23}$$

The system may be unstable, but it may satisfy the steady state at each sampling time with a few assumptions.

Steady State Parameterization

For a nominal state, we can characterize the steady state (Muske, 1997):

$$(I - A)x_s = Bu_s \tag{24}$$

Equation 24 may have eigenvalues in matrix A, so it is not possible sometimes to parameterize the input. The partitions of M are M_x and M_u and they were considered as an orthogonal matrix:

$$\begin{bmatrix} x_s \\ u_s \end{bmatrix} = M\theta \begin{bmatrix} M_x \\ M_u \end{bmatrix} \theta \tag{25}$$

Finite Prediction Horizon

Assuming [$P \in (0, \infty)$] MPC compromises addressing a finite prediction horizon over a control problem at sample time k. After this, the original input vector of the optimal grouping is applied to the framework in non-trivial time considered for the computation of an open loop system (Dekkata and Sun Yi, 2019). If the dynamics are given below, refer to Eqs. 26-27, assuming the plant is linear:

$$\dot{x}^p(t) = A^p(t)x^p(t) + B^p(t)u(t) \tag{26}$$

$$y(t) = C^p(t)x^p(t) + D^p(t)u(t) \tag{27}$$

The system is assumed to be time-invariant (LTI system), constant if all the matrices $A^p(t)$, $B^p(t)$, $C^p(t)$, and $D^p(t)$ are constant. But at least one or more than one matrix varies in time; it is assumed to be the time-invariant system.

State Reduction

For Husky, yaw stability control, the tire forces must show the lagged characteristics of the prediction model. The MPC-based yaw stability control calculations cause a critical computational weight in finding the ideal arrangements. The desired yaw rate is generated by the calculated bicycle model based on the linear approximation. Husky lateral front and rear tire models are based on the bicycle model to plan Husky behavior (Li *et al*., 2009):

$$w_b = \dot{\theta}_b \tag{28}$$

$$\dot{\theta}_r = k_r(s)\dot{s} \tag{29}$$

Linearizing the error dynamics around the reference path X_r. Assuming the unmanned ground vehicle Husky framework elements are linear time invariant over a specific vehicle condition. In this research, distinguishing the linear model by breaking down and analyzing the input and output data information tests from a straight relapse viewpoint. This approach could compute the vehicle dynamics and use the optimization algorithm with a conjugate gradient (Rakovic *et al*., 2005).

Linear model:

$$\begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\alpha}_e \\ \dot{\theta}_e \end{bmatrix} = \begin{bmatrix} 0 & k_d u_r & 0 & 0 \\ -k_d u_r & 0 & u_r & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_e \\ y_e \\ \alpha_e \\ \theta_e \end{bmatrix} + \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \tag{30}$$

Constraints

Linearizing the non-linear model with states around the stability point to produce the linear model. During this research, uncovered the assessed linear model and were able to get a final form of the algorithm and differentiate it from the linear model which was obtained (Dekkata *et al*., 2020).

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} \dot{s} + u_R \sin\alpha_e \\ \phi u_R - k_r(s)\dot{s} \\ w_m - w_b \end{bmatrix} \tag{31}$$

Husky's rotation and translation are controlled separately because of omnidirectional; and rewrite the equations as below, refer to Eq. 32 as the decoupling translation and refer to Eq. 33 for rotation:

$$X(t) = f(X(t), U(t)) \tag{32}$$

$$X(0) = X_0 \tag{33}$$

$$X(t) = \begin{bmatrix} x_m & y_m & \theta_m \end{bmatrix}^T \tag{34}$$

$$U(t) = \begin{bmatrix} u_m & v_m & w_m \end{bmatrix}^T \tag{35}$$

$X(t)$ is the state vector and $U(t)$ is the vector of Husky velocities in the body frame. The Husky's front wheels' center linear speed and rear wheels' center are relative to the ground surface. Husky wheel radius times the angular velocity gives us the tangential velocity of each wheel. For Husky's body frame, Husky can move forward or backward with only one axis; as the kinematic model is being used, velocity in a fixed structure is essential:

$$\begin{bmatrix} \dot{x}_m \\ \dot{y}_m \\ \dot{\theta}_m \end{bmatrix} = \begin{bmatrix} \cos\theta_m & -\sin\theta_m & 0 \\ -\sin\theta_m & \cos\theta_m & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_m \\ v_m \\ w_m \end{bmatrix} \tag{36}$$

$$\begin{bmatrix} \dot{x}_m \\ \dot{y}_m \\ \dot{\theta}_m \end{bmatrix} = \begin{bmatrix} u_m \\ v_m \\ w_m \end{bmatrix} \tag{37}$$

$$\begin{bmatrix} \dot{x}_m \\ \dot{y}_m \\ \dot{\theta}_t \\ \dot{\theta}_m \end{bmatrix} = \begin{bmatrix} u_R \cos\theta_t \\ u_R \sin\theta_t \\ \phi u_R \\ w_m \end{bmatrix} \tag{38}$$

Change of acceleration with the help of the rotation matrix is achieved in the fixed frame. The difference between the estimated speed and desired speed is the error signal given to the controller to generate the required torque to send the wheels to meet the desired velocity (Muktadir *et al*., 2022). Determining the curvature and the Husky translation velocities, refer to Eq. 39. Experiencing errors in the steady state in the yaw angle. So, minimizing the lateral deviation to zero eliminates the yaw angle's steady state error. The plant model minimizes the steady state error to control Husky's longitudinal and lateral motion:

$$\begin{bmatrix} u_m \\ v_m \end{bmatrix} = \begin{bmatrix} u_R \cos(\theta_t - \theta_m) \\ u_R \sin(\theta_t - \theta_m) \end{bmatrix} \tag{39}$$

Concepts of Prediction

Through model equations, the estimation of the unknown state $x(1|k)$, is determined by the current state $x(k|k)$. The size of the prediction issue presented in this manner increments straight with the number of estimations. For a prediction method to be computationally possible, the option to bind the number of factors is assessed. The group assessment issue can be adjusted to utilize a fixed size moving horizon state in which several predictions are based on the plant size (Rawlings *et al*., 1994):

$$\begin{aligned} &\min_{x^e(k-m+1|k)} J\left(\bar{p}, x(k-m+1|k)\right) \\ &\equiv \frac{1}{2}\Big[\left(x^e(k-m)+1|k\right)' P(k-m+1|k) \\ &*x^e(k-m+1|k) + \sum_{l=k-m+1}^{k} v(l|k)' \bar{R}^{-1} v(l|k)\Big] \end{aligned} \tag{40}$$

Subject to:

$$v(l|k) = y_l^p - g\left(x(l|k), \bar{p}\right), l = k-m+1,...,k \tag{41}$$

$$\dot{x}(l|k) = f\left(x(t|k), u_{l-1}, \bar{p}\right), l = k-m+1,...,k \tag{42}$$

Initially, the prediction is the number of measurements modeled to estimate the horizon's size at the next step. Predicting the least squares $(k - m + 1|k)$ at $t = t_{k-m+1}$ given at time t_{k-1}. The weighting matrices predict the confidence in the state $(k - m + 1|k - 1)$ with the conditional inverse covariance matrix. Repeating the procedure at each time step, with all the future time's probabilistic interpretation for the optimization of the model at constant covariance of the prediction model $(k - m + 1|k) - x(k - m + 1|k - 1)$, then we obtain the $P^{-1}(k - m + 2|k)$, from $P^{-1}(k - m + 1|k - 1)$ using the filter technique (Ferreau *et al*., 2017).

The size of the assessment issue presented in this manner increments with the number of estimations. For an assessment strategy to be computationally possible several factors are assessed (Diehl, 2021). The number of estimates based on the size of the optimization remains constant for the moving horizon problem over at a time t_k and $m - 1$, as the horizon size. A linearized model is utilized when the proportionality between Extended Kalman Filter (EKF) method and the moving least square calculation method is evaluated. At a moment when $m = 1$, the state conditions try not to show up in the control problem and the issue turns into a direct least squares problem. The acquired arrangement relates to the estimation adjustment step of the EKF. In this manner, the moving horizon estimator based on EKF is identical at point $m = 1$, (Ferreau *et al*., 2012). Moreover, the prediction horizon size is solitary with extra tuning boundaries other than those utilized in the EKF method.

Controller Tuning

Controller tuning is done by logging various vehicle kinematic data sets during testing, such as acceleration

DOI: 10.3844/jmrsp.2023.27.41

torque inputs and yaw rate (Kong *et al*., 2015). After careful analysis of collected data and documentation, changes are then applied to the control parameters and tested again. If drastic changes are made to the control parameters or structure, they are pushed through simulation first. All control inputs are carefully observed before implementation, which helps limit any surprise responses that may occur. After tuning, following control input before the performance tuning by sending minimum and maximum command values are further suggested abnormalities. Software protections such as saturation blocks are added to enforce the control output limits. Repeating this procedure several times is an ongoing approach to generate a smooth trajectory for Husky.

Process Modelling

LiDAR Measurements

One VLP-16 from Velodyne is mounted on top of the Husky parallel to the ground surface with a 360° view. This coverage will allow the Husky to cover close objects approaching from all directions and for more precise localization; with the localization information, the Husky's current position can also be detected along with the landmarks around.

Figure 4, the architecture of LiDAR ground surface estimation, from LiDAR and IMU, the azimuth and relative position changes are integrated at a rate of 5 hz. The measurement model is given. This model is used when the LiDAR and IMU are loosely coupled:

$$\left[\Delta x_{lidar} - \Delta x_{ins} \Delta y_{lidar} - \Delta y_{ins} \Delta A_{lidar} \Delta A_{ins}\right] = I_{3\times 3}\left[\delta\Delta_x \, \delta\Delta_y \, \delta\Delta_z\right] \quad (43)$$

where, Δx_{ins}, Δy_{ins}, and ΔA_{ins} are:

$$\Delta x_{ins} = v \cos p \sin \Delta A * T \quad (44)$$

$$\Delta y_{ins} = v \cos p \cos \Delta A * T \quad (45)$$

$$\Delta A_{ins} = \omega_z^l T \quad (46)$$

where:

ω_z^l = The angular velocity along the vertical axis
T = The sampling time

The angular velocity along the vertical axis is projected in the horizontal plane. LiDAR is mounted on the Husky and is connected to the computing platform using the ethernet cable.

LiDAR-Based System Components

Entities are detected in the surrounding space and are segmented using range information provided by LiDAR. On 2D laser range images, several possible segmentation methods to perform the segmentation stage a linear Kalman Filter (KF) based method have been used. From the extracted segments, tracking and data association techniques are performed in the referential laser system.

Figure 5, the block diagram illustration of LiDAR-based detection, and the detected elements are segmented and classified using filtering and feature detection. Tracker prediction under tracking is considered to evolve according to a stochastic dynamic model driven by process noise. The state and measurement uncertainties are considered white Gaussian noise with zero mean and known covariance matrices. Object tracking using a multi-independent KF strategy is performed in cartesian space (Abbas *et al*., 2017).

Figure 6 shows, a LiDAR map builder has created and loops through the point cloud array and updates the map with a new LiDAR frame and updates the view display.

Detecting and Tracking Objects

Feature extraction from the segments under tracking, probable objects' interest, and a feature vector is extracted to perform the classification stage. This feature vector has the following components:

*f*1) Segment centroid: *x*

*f*2) Normalized cartesian dimension: This feature corresponds to the root mean square of the segment width *(ΔX)* and length *(ΔY)* dimensions:

$$f2 = \sqrt{\Delta X^2 + \Delta Y^2} \quad (47)$$

*f*3) Internal standard deviation: Denotes the standard deviation of the range points *(m)* for the segmented centroid:

$$f3 = \sqrt{\frac{1}{n-1}\sum_j^1 \left\| r_n - \bar{x} \right\|} \quad (48)$$

*f*4) Radius: Denotes the radius of the circle extracted from the segment points. The used circle fitting is based on Guivant's method

*f*5) Deviation: The mean average deviation from the median feature is expressed by:

$$f5 = \frac{1}{n-1}\sum_j^1 \left\| r_n - \bar{x} \right\| \quad (49)$$

DOI: 10.3844/jmrsp.2023.27.41

LiDAR-Based Classifier

In the classifier, which was implemented based on the data from the LiDAR. Each object category is modeled by a finite distribution whose parameters were estimated during supervised training. The decision rule is based on the posterior probability that each object belongs to the class interest. The objects are modeled by a weighted combination of Gaussian probability, referred to as Gaussian of the mixture model describing the class (object category).

Figure 7 shows, the detection, and tracking of multiple objects in the frame. The probability capacities are converted into more extensive planning, which keeps communicating the object's undoubted location.

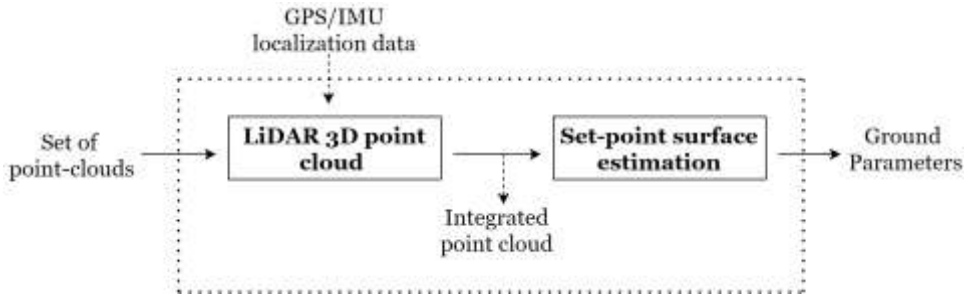

Fig. 4: LiDAR ground surface estimation

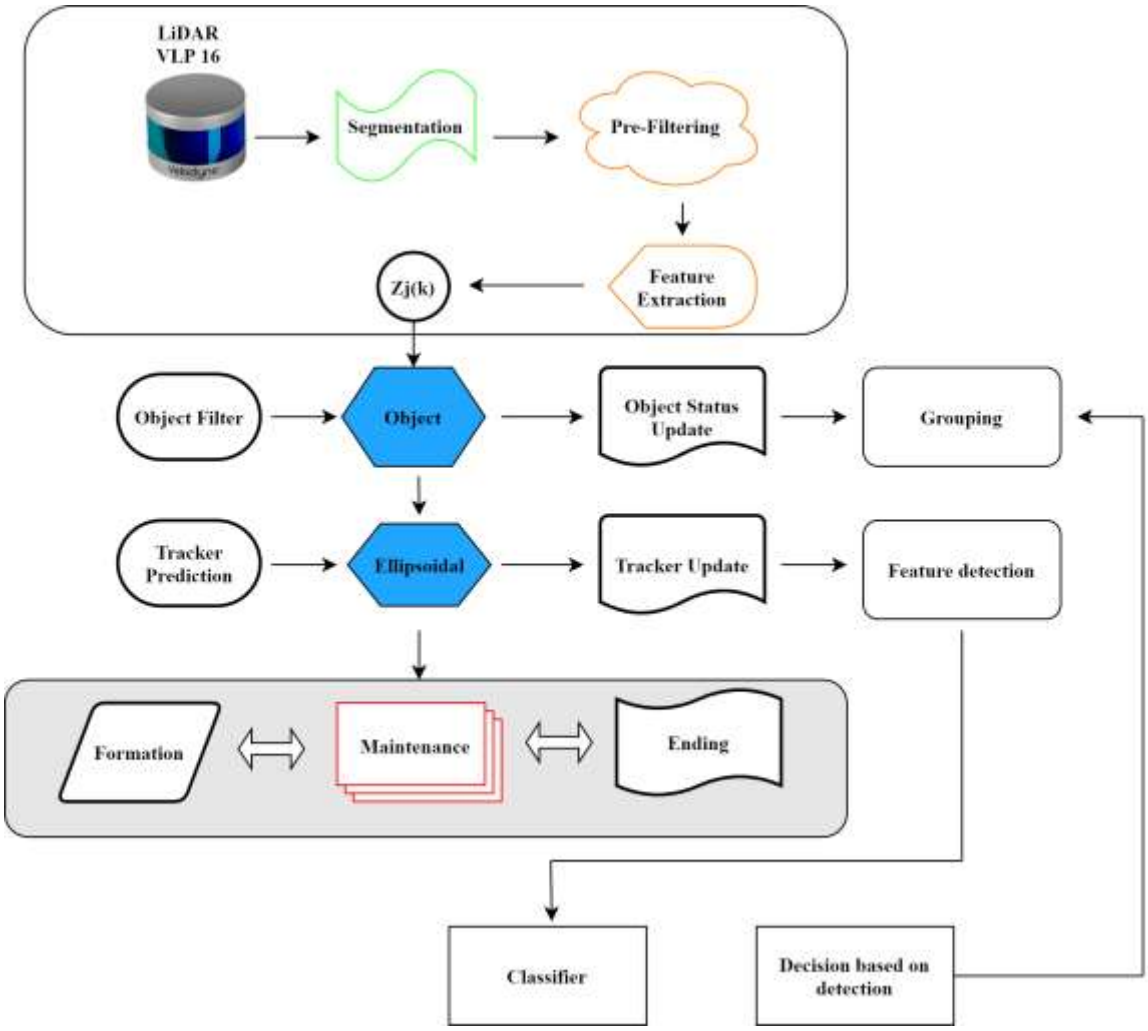

Fig. 5: Block diagram illustration of LiDAR-based detection

DOI: 10.3844/jmrsp.2023.27.41

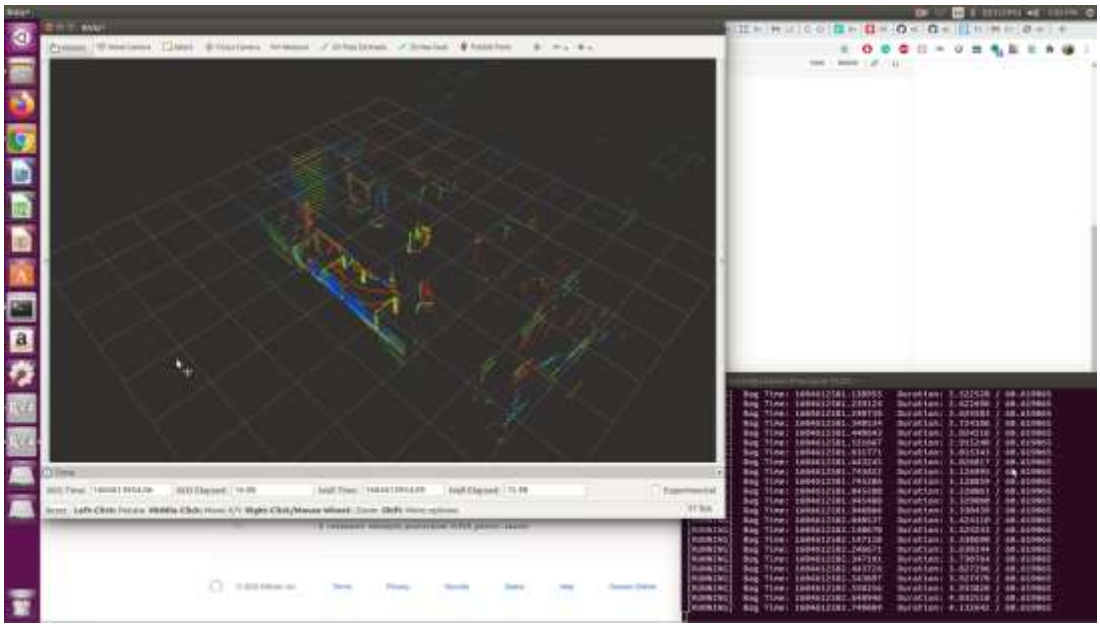

Fig. 6: LiDAR VLP-16 point cloud

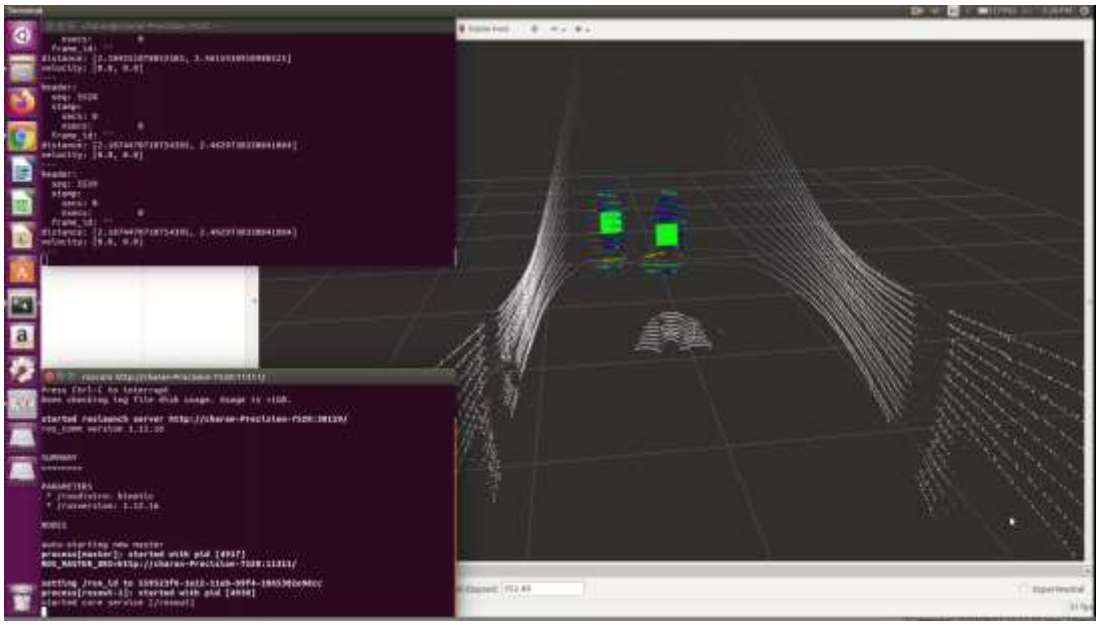

Fig. 7: Detecting multiple objects

Segment Clusters

The segment clusters which define an object are a cloud of range points. The centroid of each cluster calculates all the characteristic points i.e., with respect to the centroid of the cluster, the object's behavior is described in the kinematic model. Intrinsically connected with tracking, data association is performed considering two situations. Segment to segment is the process of associating detected segments with the other segments (non-classified objects) in the current scan (Garfo *et al*., 2020).

Figure 8, the LiDAR 3D map is the association of observed segments with existing objects. The first situation occurs when one or more current segments are probably related to a past segment under tracking. To deal with this situation, a combination of rectangle gate and feature matching techniques are used. The second data association problem is the observation of tracker association, which is solved in a specific manner that accounts for object classification.

DOI: 10.3844/jmrsp.2023.27.41

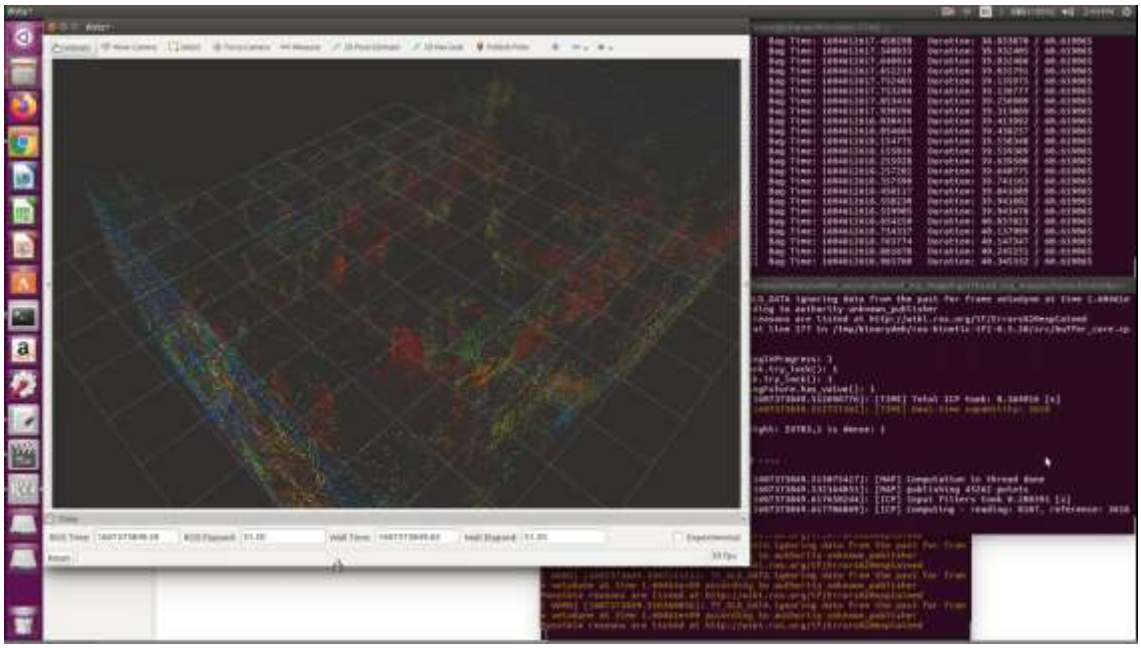

Fig. 8: LiDAR 3D map

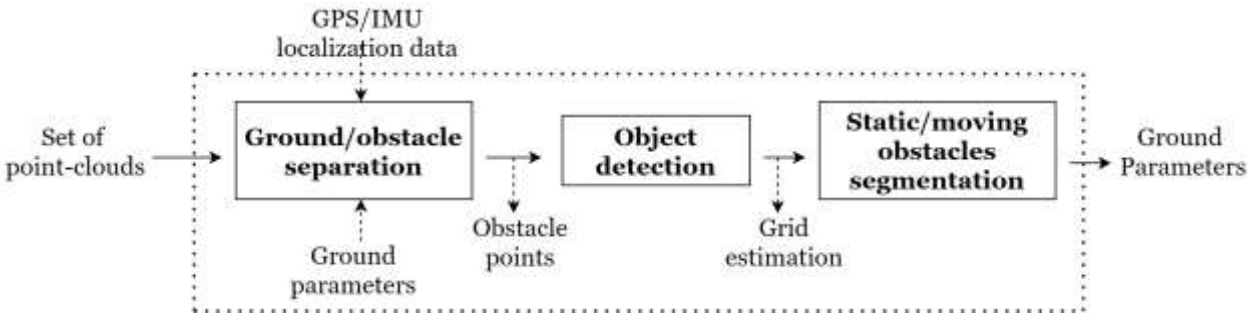

Fig. 9: LiDAR static and moving objects detection

Figure 9, using GPS coordinates to localize the Husky, based on the high-level decisions and a path is generated (whether to go straight or turn), following immediate waypoints will be generated. In a segment-to-object tracker, the maintenance process is the association of observed segments with existing objects. The first situation occurs when one or more current segments are probably related to a past segment under tracking. To deal with this situation, a combination of rectangle gate and feature matching techniques are used. The second data association problem is the observation of tracker association, which is solved in a specific manner that accounts for object classification (Asvadi *et al*., 2016).

Results and Discussion

Simulation Results

Figure 10, the efficiency of the proposed model predictive control paradigm is tested through simulation. The automated driving toolbox from math works is used for performing the simulation. The Husky under control is assumed to be 1 m long. The actual state and input state of Husky at each time step *t* is defined in equations below, refer to Eqs. 50-51 (Dekkata *et al*., 2022):

$$y_c(t) = y_e(t) + y_r(t) \tag{50}$$

$$v_c(t) = v_e(t) + v_r(t) \tag{51}$$

Controlled by the Husky's nominal input, the nominal disturbance of free state and nominal input of the Husky is evaluated using the below equations, refer to Eqs. 52-53:

$$\bar{y}_c(t) = \bar{y}_e(t) + y_r(t) \tag{52}$$

$$\bar{v}_c(t) = \bar{v}_e(t) + v_r(t) \tag{53}$$

Based on the plant's feedback, Husky's reference trajectory is calculated using the actual and free state:

$$x_r(t) = 6.9 + 0.55\sin\left(\frac{2\pi t}{3.8}\right) \quad (54)$$

$$y_r(t) = 4.6 + 0.55\sin\left(\frac{4\pi t}{3.8}\right) \quad (55)$$

$$\theta_r(t) = arctan2\left(\dot{x}_r(t), \dot{y}_r(t)\right) \quad (56)$$

Comparing Simulation and Implementation Results

Figure 11, the real-time longitudinal movement follows the simulation results closely. The time-based tracking performance of the Husky in simulation is compared with the real-time implementation's ideal control strategy. The MPC could stabilize the Husky longitudinal motion with a more modest value of the corrective yaw moment.

Figure 12 shows that the real-time lateral movement follows the simulation closely. The MPC could stabilize the Husky lateral motion with a more modest value of the corrective yaw movement.

The Ark bridge map generates a set of waypoints as the desired trajectory. All the waypoints in the reference trajectory are fitted in a polynomial order. For a curvy path, in this study, a 3rd degree polynomial worked well. According to the Husky position and orientation, applying a transformation to convert the map coordinate system to the Husky coordinate system. Transforming the map coordinate system would help compute the errors because the initial state vector considers position and orientation zero.

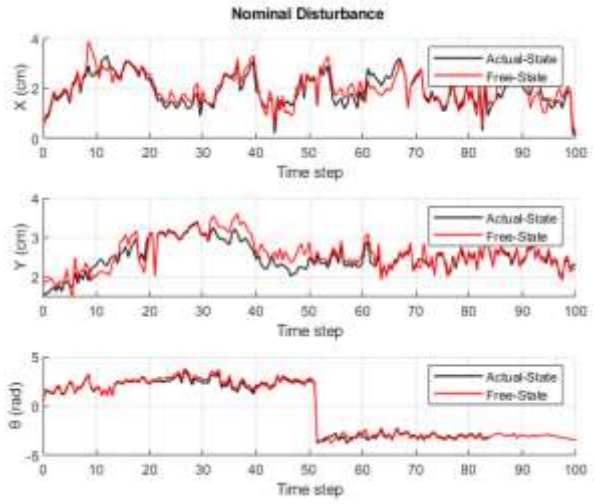

Fig. 10: Husky simulation results

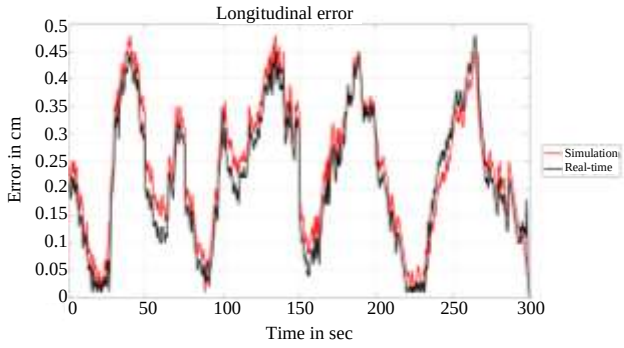

Fig. 11: Longitudinal error to the reference trajectory

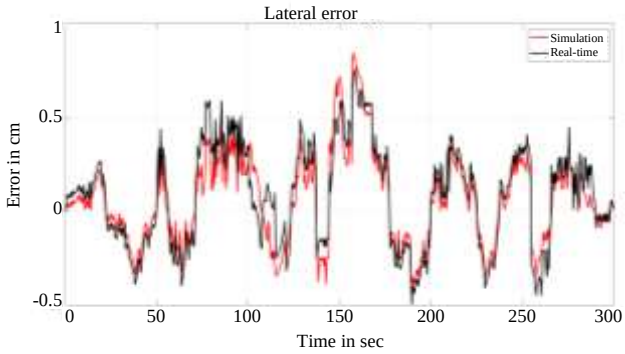

Fig. 12: Lateral error to the reference trajectory

Conclusion

In this study, an advanced MPC method was developed for autonomous navigation systems combined with control theories. The goal is to design an advanced MPC for Husky A200 to navigate autonomously based on the Husky dynamics. The Husky A200 ground robot can navigate autonomously with waypoints, and it is tested indoors in unstructured conditions without deduced data about the obstructions. This research also presents a new obstacle avoidance algorithm calculation MPC based, which improves longitudinal speed and guiding point while exploring the region, Husky securely yet as quickly as possible to the target area. By reducing the cross-track error and orientation error in the active set method, the MPC is optimized. The best possible route is extracted, and the trajectory is generated based on Husky's current events and position. Husky can avoid obstacles for optimal path planning and target following. The proposed algorithm for Husky A200 is tested indoors and compared the results with the simulation results which are plotted using MATLAB and Gazebo. A novel simulator package is designed by modeling the Husky using RVIZ and Gazebo. Software protections such as saturation blocks are added to enforce the control output limits, repeating this procedure several times during the approach generated

a smooth trajectory for Husky. For MPC's short-term optimization, an optimized control signal from a linear framework is utilized for a linear quadratic controller. The real-time longitudinal movement follows the simulation closely. Based on the plant's feedback Husky's reference trajectory is calculated using the actual state and free state.

Acknowledgment

This study was supported by the Center for Advanced Transportation Mobility (CATM) at North Carolina A&T State University.

Funding Information

The authors gratefully acknowledge the support of the center for advanced transport mobility and the department of energy Minority Serving Institution Partnership Program (MSIPP) managed by Savannah River national laboratory for providing funds to pursue this research.

Author's Contributions

Sai Charan Dekkata: Designed and developed the MPC and object detection and avoidance algorithm for the unmanned ground vehicle Husky A200.

Sun Yi: Supported MPC designed, reviewed the journal, and supervised the research.

M. A. Muktadir and Selorm Garfo: Proofread the manuscript.

Ethics

This article contains information which is published in my Ph.D. dissertation. The corresponding author confirms that all the other authors have read and approved the manuscript and that no ethical issues are involved.

References

Abbas, M. A., Milman, R., & Eklund, J. M. (2017). Obstacle avoidance in real time with nonlinear model predictive control of autonomous vehicles. *Canadian Journal of Electrical and Computer Engineering*, *40*(1), 12-22. https://doi.org/10.1109/CJECE.2016.2609803

Asvadi, A., Premebida, C., Peixoto, P., & Nunes, U. (2016). 3D Lidar-based static and moving obstacle detection in driving environments: An approach based on voxels and multi region ground planes. *Robotics and Autonomous Systems*, *83*, 299-311. https://doi.org/10.1016/j.robot.2016.06.007

Bock, H. G., & Plitt, K. J. (1984). A multiple shooting algorithm for direct solution of optimal control problems. *IFAC Proceedings Volumes*, *17*(2), 1603-1608. https://doi.org/10.1016/S1474-6670(17)61205-9

Boyd, S., Boyd, S. P., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press. https://web.stanford.edu/~boyd/cvxbook/

Dekkata, S. C. (2018). *Steering and adaptive cruise control for autonomous vehicles using model predictive control* (Doctoral dissertation, North Carolina Agricultural and Technical State University.

Dekkata, S. C. (2021). *Model Predictive Control for Unmanned Ground Vehicles Using Robot Operating System* (Doctoral dissertation, North Carolina Agricultural and Technical State University).

Dekkata, S. C., & Yi, S. (2019). Improved steering and adaptive cruise control for autonomous vehicles using model predictive control. *Journal of Mechatronics and Robotics*, *3*(1), 378-388. https://doi.org/10.3844/jmrsp.2019.378.388

Dekkata, S. C., Okore-Hanson, T., Yi, S., Hamoush, S., Seong, Y., & Plummer, J. (2020, July). Autonomous Navigation and Control of UGVs' in Nuclear Power Plants-20381. WM Symposia, Inc., PO Box 27646, 85285-7646 Tempe, AZ (United States). https://www.osti.gov/biblio/23028007

Dekkata, S. C., Yi, S., Muktadir, M. A., Garfo, S., Li, X., & Tereda, A. A. (2022). Improved Model Predictive Control System Design and Implementation for Unmanned Ground Vehicles. https://doi.org/10.3844/jmrsp.2022.90.105

Diehl, M. (2021). Optimization algorithms for model predictive control. In *Encyclopedia of Systems and Control* (pp. 1619-1626). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-44184-5_9

Diehl, M., Bock, H. G., Schlöder, J. P., Findeisen, R., Nagy, Z., & Allgöwer, F. (2002). Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, *12*(4), 577-585. https://doi.org/10.1016/S0959-1524(01)00023-3

Ferreau, H. J., Almér, S., Verschueren, R., Diehl, M., Frick, D., Domahidi, A., ... & Jones, C. (2017). Embedded optimization methods for industrial automatic control. *IFAC-Papers OnLine*, *50*(1), 13194-13209. https://doi.org/10.1016/j.ifacol.2017.08.1946

Ferreau, H. J., Kraus, T., Vukov, M., Saeys, W., & Diehl, M. (2012, December). High-speed moving horizon estimation based on automatic code generation. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)* (pp. 687-692). IEEE. https://doi.org/10.1109/CDC.2012.6426428

Gao, Y., Lin, T., Borrelli, F., Tseng, E., & Hrovat, D. (2010, January). Predictive control of autonomous ground vehicles with obstacle avoidance on slippery roads. In *Dynamic Systems and Control Conference* (Vol. *44175*, pp. 265-272). https://doi.org/10.1115/DSCC2010-4263

Gardezi, M. S. M., & Hasan, A. (2018). Machine learning based adaptive prediction horizon in finite control set model predictive control. *IEEE Access*, *6*, 32392-32400. https://doi.org/10.1109/ACCESS.2018.2839519

Garfo, S., Muktadir, M. A., & Yi, S. (2020). Defect detection on 3D print products and in concrete structures using image processing and convolution neural network. *Journal of Mechatronics and Robotics*, *4*(1), 74-84. https://doi.org/10.3844/jmrsp.2020.74.84

Goodwin. G. C. (2005). Constrained Control and Estimation. Heidelberg: Springer-Verlag ISBN: 185233-548-3. https://books.google.com.sb/books?id=Rp2F_StqjAEC&printsec=frontcover#v=onepage&q&f=false

Grüne, L. (2012). NMPC without terminal constraints. *IFAC Proceedings Volumes*, *45*(17), 1-13. https://doi.org/10.3182/20120823-5-NL-3013.00030

Kirches, C. (2011). *Fast numerical methods for mixed-integer nonlinear model-predictive control*. Wiesbaden, Germany: Vieweg + Teubner Verlag. https://doi.org/10.1007/978-3-8348-8202-8

Kong, J., Pfeiffer, M., Schildbach, G., & Borrelli, F. (2015, June). Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE intelligent vehicles symposium (IV)* (pp. 1094-1099). IEEE. https://doi.org/10.1109/IVS.2015.7225830

Li, M., Imou, K., Wakabayashi, K., & Yokoyama, S. (2009). Review of research on agricultural vehicle autonomous guidance. *International Journal of Agricultural and Biological Engineering*, *2*(3), 1-16. http://ijabe.org/index.php/ijabe/article/view/160

Mayne, D. Q., & Langson, W. (2001). Robustifying model predictive control of constrained linear systems. *Electronics Letters*, *37*(23), 1422-1423. https://digital-library.theiet.org/content/journals/10.1049/el_20010951

Mayne, D. Q., & Michalska, H. (1988, December). Receding horizon control of nonlinear systems. In *Proceedings of the 27th IEEE Conference on Decision and Control* (pp. 464-465). IEEE. https://doi.org/10.1109/CDC.1988.194354

Mayne, D. Q., Rawlings, J. B., Rao, C. V., & Scokaert, P. O. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, *36*(6), 789-814. https://doi.org/10.1016/S0005-1098(99)00214-9

Muktadir, M. A., & Yi, S. (2021, July). Machine Vision-Based Detection of Surface Defects of 3D-Printed Objects. In *2021 ASEE Virtual Annual Conference Content Access*. https://peer.asee.org/machine-vision-based-detection-of-surface-defects-of-3d-printed-objects

Muktadir. M A., Yi, S., Hamoush, S., Garfo, S., Dekkata, S. C., Li, X., Tereda, A., Mckee, R., Brown, K., & Klawah, N. (2022). Uncrewed Ground Vehicles (UGVs) and Nature-Inspired Designed Robot DIGIT and SPOT: A Review. *American Journal of Engineering and Applies Sciences*, *15*(4), 274-287 https://doi.org/10.3844/ajeassp.2022.274.287

Muraleedharan, A., Okuda, H., & Suzuki, T. (2021). Real-time implementation of randomized model predictive control for autonomous driving. *IEEE Transactions on Intelligent Vehicles*, *7*(1), 11-20. https://doi.org/10.1109/TIV.2021.3062730

Muske, K. R. (1997, June). Steady-state target optimization in linear model predictive control. In *Proceedings of the 1997 American control conference (Cat. No. 97CH36041)* (Vol. *6*, pp. 3597-3601). IEEE. https://doi.org/10.1109/ACC.1997.609493

Rakovic, S. V., Kouramas, K. I., Kerrigan, E. C., Allwright, J. C., & Mayne, D. Q. (2005). The minimal robust positively invariant set for linear difference inclusions and its robust positively invariant approximations. *Automatica*.

Rawlings, J. B., Meadows, E. S., & Muske, K. R. (1994). Nonlinear model predictive control: A tutorial and survey. *IFAC Proceedings Volumes*, *27*(2), 185-197. https://doi.org/10.1016/S1474-6670(17)48151-1

Richards, A. G. (2005). *Robust constrained model predictive control* (Doctoral dissertation, Massachusetts Institute of Technology). https://dspace.mit.edu/handle/1721.1/28914

Sutton, G. J., & Bitmead, R. R. (2000). Performance and computational implementation of nonlinear model predictive control on a submarine. In *Nonlinear Model Predictive Control* (pp. 461-472). Birkhäuser Basel. https://doi.org/10.1007/978-3-0348-8407-5_27

Vougioukas, S., Arvanitis, K., & Sigrimis, N. (2007, July). A nonlinear model predictive tracking controller for agricultural vehicles. In *2007 European Control Conference (ECC)* (pp. 4937-4943). IEEE. https://doi.org/10.23919/ECC.2007.7068764

Wang, Y., & Boyd, S. (2009). Fast model predictive control using online optimization. *IEEE Transactions on Control Systems Technology*, *18*(2), 267-278. https://doi.org/10.1109/TCST.2009.2017934

Zeilinger, M. N. (2011). Real-time Model Predictive, Published dissertation in partial fulfillment of the requirements for the degree Doctor of Philosophy, ETH ZURICH, Keplerstrabe, Stuttgart, Germany. https://doi.org/10.3929/ethz-a-6619878