Journal of Mechatronics and Robotics

Original Research Paper

Improved Model Predictive Control System Design and Implementation for Unmanned Ground Vehicles

Sai Charan Dekkata, Sun Yi, MA Muktadir, Selorm Garfo, Xingguang Li and Amanuel Abrdo Tereda

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

Article history
Received: 28-09-2022
Revised: 10-10-2022
Accepted: 13-10-2022

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

Abstract: Autonomous ground robots autonomously are being used in the places where it is very hazardous for human beings to reach and operate, such as nuclear power plants and chemical industries. The aim of the research presented here is to develop a control system that enables such ground robots navigate autonomously with various sensors as the depth camera, 2D scanning laser, 3D Lidar, GPS, and IMU. The controller uses the current position measured using the sensors on the Husky A200, given the waypoints of the destination. Then it calculates the best possible route based on the recent events provided using IMU data and GPS. The Model Predictive Control (MPC) improves the robot's motion, by using a path planner for the robot's trajectory generation. The use of global reference frame waypoints is planned to create the appropriate path and the actions required to follow the motion planner's direction. The path planner depends on the active sensor data such as locations and size of obstacles. Then, a feasible path is generated based on the sensor data. The desired trajectory consists of a set of waypoints fit in a 3rd-order polynomial. They determine the path's feasibility for the ground robot's dynamics and a series of points generated with a certain velocity and acceleration profile. The MPC adjusts the robot's lateral, longitudinal, yaw motions and approximates a continuous trajectory with discrete paths to command behaviors. The kinematic model of a robot, Husky is used as the dynamic model for transient and steady-state characteristics. The camera captures the images and other types of data processed through the computational framework used to build machine learning models. TensorFlow is used for deep learning and to identify and classify various objects around the Husky. This research has limitations such as using the linear dynamic model as the LQR method. Also on vehicle models, the vehicle model considered in this research considers a constant value to describe the slope in the most linear region. Detailed discussion on MPC development with a major system design factor has been emphasized with logical steps in MPC.

Keywords: Model Predictive Control, Unmanned Ground Robot, Autonomous Navigation, Trajectory Generation

Introduction

The Model Predictive Control (MPC) is a control method for optimization strategy in a feedback control system (Xie and Fierro, 2008). It runs a set of forecasts of the model forward in time, optimizing control input for different actuation strategies. In the immediately following control action, it reinitializes the optimization to find the subsequent control inputs. The objective is to convey a custom-made MPC law to the framework and equipment viable and to meet the computational necessities, specifically regarding the continuous limitations (Carlos and Garcia, 1989).

The ideal MPC law gives closed-loop stability and requirement fulfillment consistently under certain presumptions on the complex arrangement (Yoon *et al*., 2009). The computational necessities depend on calculating the optimal control activity by addressing a finite horizon optimal control problem. For example, it is needed to optimize control over every new time step and reassess the optimizing control input over moving horizons (Zanon *et al*., 2013). It depends on the robust control activity calculation by addressing a compelled ideal control issue for the plant's present status at each sample time (Bahadorian *et al*., 2011). The authors introduced an objective in delicate obliged MPC with a focal point of convergence for developing continuous and sensitive obliged systems which are consistently relinquishing in common articulations (Dekkata and Yi, 2019).

Thinking about the computational necessities of the MPC on an additional room for the estimation, time is considered for the utilization of interest. Highspeed applications, for instance, force problematic ongoing limitations on the arrangement of the MPC issue (Ferreau *et al*., 2008). For example, a breaking point on the calculation time is accessible to process the control input. A higher perspective here is that instead of doing the control streamlining one time disconnected to use with a linear quadratic regulator. Consistently doing this enhancement at each framework's progression (Kuhne *et al*., 2004), adjusting the Husky' s controller behavior if framework begins to veer off or if the elements begin to change. MPC has driven a considerable measure of interest in system identification since it depends on a framework model. Keen on getting great system models that can be utilized due to the model's predictive control force, so the system' s non-linear and linear models can handle (Maynea *et al*., 2005).

Figure 1 husky A200 from Clear path Robotics is mounted with LiDAR in the image to the right. The LiDAR point cloud data is visualized with laptop connected to the LiDAR. The work presented in this study mainly concentrates on the control system for unmanned ground vehicles and designing an effective control system for Husky's autonomous navigation. The unmanned ground robot Husky A200 is equipped with Velodyne 3D Lidar, Intel depth camera (D435i), Hokuyo 2D scanning laser, M1125 axis camera, Global Positioning System (GPS), and Inertial Measurement Unit (IMU). In door, GPS is unreliable and IMU helps navigate through waypoints. The Lidar data is to identify objects and M1125 axis camera is to identify humans using the TensorFlow models and the Intel depth camera D435i to identify static and dynamic obstacles more precisely and determine the objects' time-varying position (Mayne *et al*., 2009). The Husky' s relative position gives the objects' position in the local frame and estimates detected objects attitude for the Husky' s absolute positioning.

Figure 2 the control design system architecture of the Husky is illustrated. Most sensors communicate through the ethernet cable and all the sensors are connected to a switch on-board.

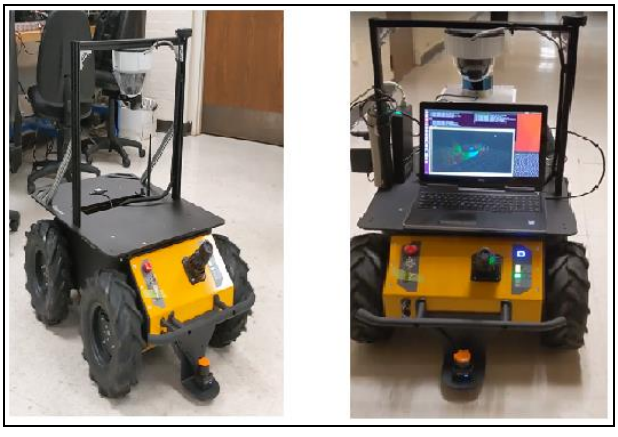

Fig. 1: Husky A200-clear path robotics

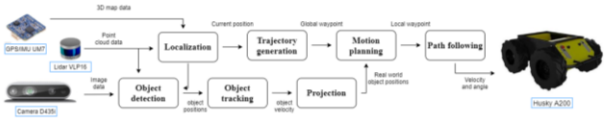

Fig. 2: Control design architecture

Materials and Methods

Vehicle Modeling

The use of the Kinematic model helps simplify the Husky physics and consider a set of models suitable for model-based control design. The kinematic model describes Husky' s kinematic motion and the dynamic model of the Husky is designed to consider the transient and steady-state characteristics of the Husky (Ko and Song, 2010).

Figure 3 the vehicle coordinate system calculates the force acting on the Husky's center of gravity with the body frame of references *x*, *y*, and *z* are longitudinal, lateral, and vertical axes, respectively. The forces acting on the center of gravity of the Husky are formulated as longitudinal and lateral, F_x and F_y, and the rotating moment of the z-axis is M_z.

Rigid body dynamic equations are:

$$m\ddot{x} = F_x + m\dot{y}\dot{\varphi}, \tag{1}$$

$$m\ddot{x} = Fy - m\dot{y}\dot{\varphi}, \tag{2}$$

$$I_z\ddot{\varphi} = M_z, \tag{3}$$

where:
I = The Husky's moment of inertia about the z-axis
m = The vehicle mass
F_x = Longitudinal
F_y = Lateral

The longitudinal dynamics, lateral dynamics, yaw dynamics, and wheel dynamics are modeled. The roll dynamics, pitch dynamics, and vertical dynamics are ignored since the Husky moves at relatively low velocity (Kiencke and Nielsen, 2005):

$$\dot{X} = \dot{x}\cos\varphi - \dot{y}\sin\varphi, \tag{4}$$

$$\dot{Y} = \dot{x}\sin\varphi + \dot{y}\cos\varphi \tag{5}$$

Four-Wheel Model

Figure 4 all the forces on the Husky are illustrated the simplified vehicle dynamics model is used to develop the controller to minimize the deviations in the accelerations and the variations in the Husky velocity. Assuming constant normal tire loads, using the four-wheel model to describe the dynamics of the Husky, the dynamic model is given by:

$$F_{zf,l}, F_{zf,r}, F_{zr,l}, F_{zr,r} = constant \quad (6)$$

$$m\ddot{x} = m\dot{y}\dot{\varphi} + F_{xf,l} + F_{xf,r} + F_{xr,l} + F_{xr,r} \quad (7)$$

$$m\ddot{y} = -m\dot{x}\dot{\varphi} + F_{yf,l} + F_{yf,r} + F_{yr,l} + F_{yr,l} \quad (8)$$

$$I\ddot{\varphi} = a\left(F_{yf,l} + F_{yf,r}\right) - b\left(F_{yr,l} + F_{yr,l}\right) + c\left(-F_{xf,l} + F_{xf,r} - F_{xr,l} + F_{xr,r}\right) \quad (9)$$

where:
$F_{f,l}$ = Front left tire
$F_{f,r}$ = The front right tire
$F_{r,l}$ = The rear left tire
$F_{r,r}$ = The rear right tire

The forces are applied on the center of gravity due to all four tires' longitudinal and lateral forces:

$$F_y = F_l \ \sin\sin\delta + F_c \ \cos\cos\delta \quad (10)$$

$$F_x = F_l \ \cos\cos\delta - F_c \ \sin\sin\delta \quad (11)$$

This model incorporates the vehicle body's pitch and move, the movement of four wheels in the accelerating or down braking measure, the non-linear coupling of the vehicle body, and the unsprung mass, just as the air drag coefficient wind impact. The total vertical load of the vehicle is distributed between the front and rear wheels as (Dekkata, 2021):

$$F_{zf} = \frac{bmg}{2(a+b)} \quad (12)$$

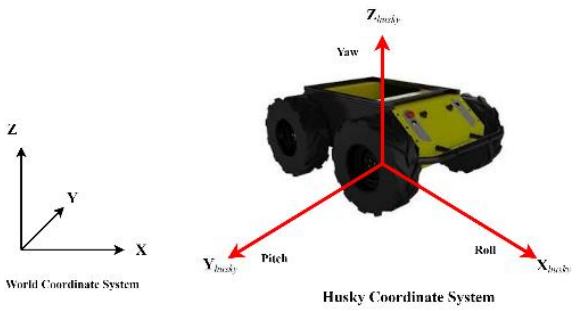

Fig. 3: Husky coordinate system

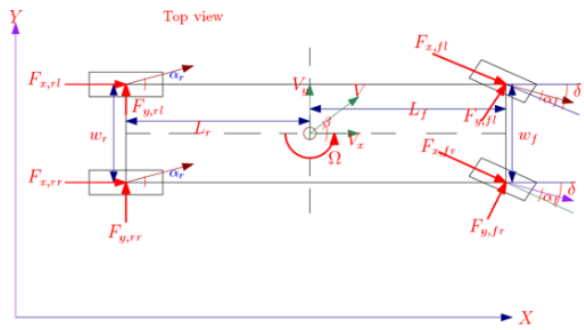

Fig. 4: Forces acting on each tire

$$F_{zr} = \frac{amg}{2(a+b)} \quad (13)$$

where:
a = The longitudinal distance between center of gravity and front-axle
b = The longitudinal distance between center of gravity and rear-axle

The model can be utilized for the examination of different moves of the four-wheel directing vehicles. Also, the previous model can be considered an extraordinary instance of this model. The compact differential equation is calculated by describing the non-linear vehicle dynamics by assuming slip ratio s and μ friction coefficient:

$$\dot{\xi} = f_{s,u}(\xi, u) \quad (14)$$

$$\eta = h(\xi) \quad (15)$$

While accelerating and down braking with tires, the Husky tires observed angular velocity doesn' t coordinate the average speed of rolling motion, implying sliding movement in rare instances. As Husky moves with much lower speeds than a regular automobile, the slip ratio is negligible:

State vectors are defined as:

$$\xi = [y, \dot{y}, \dot{x}, \varphi, \dot{\varphi}, Y, X] \quad (16)$$

Input vector is given by:

$$u = \delta_f \quad (17)$$

Model Predictive Control

Model Predictive Control depends on calculating the ideal control activity by tackling a finite horizon with an ideal control issue for the plant's present status at each sample time. Control for the Husky is designed using MPC instead of Proportional Integral plus Derivative (PID) control. The designed controller adjusts the Husky's longitudinal, lateral, and yaw motion to command behaviors. The main reason for shifting from PID to MPC is that it can optimize current time slots while taking future time slots into account to provide a smoother and more proactive control mechanism.

Model

Figure 5 the MPC schematic is illustrated. The model requires prediction and thus, logically are tackled first. To automate MPC, it is necessary the concepts in mathematical terms are evaluated first. Simple manipulation and algebra need linear models as the superposition can be used. Typically, linear models are transfer function or state-space (Mikael Eklund *et al*., 2012).

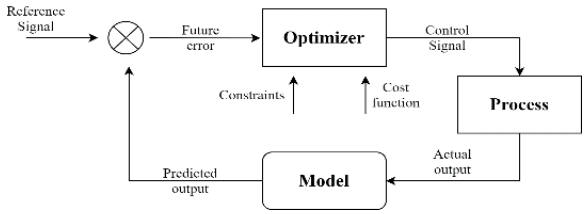

Fig. 5: Model predictive control schematic

Process Model

The optimization algorithm minimizing the cost function using the control input and obtains the optimal control action based on the current state and evaluation of PWA coefficients. This study approaches the control problem without violating the constraints low or high limits with the weighting coefficient reflecting Husky's relative motion, initially introducing the highspeed frameworks forcing on demanding ongoing requirements to calculate the MPC control law. The MPC activity is obtained by executing an advancement that is pre-computed and stored (Dekkata, 2018):

$$x_{k+1} = Ax_k + Bu_k \tag{18}$$

$$y_k = Cx_k + Du_k \tag{19}$$

Generally, the MPC models' complex dynamical behaviors require the model to minimize the difference between predicted output and desired output. It reduces to the linear quadratic regulator if there are no constraints in MPC. The succession of optimal control input is registered for an anticipated advancement of the framework model over a limited period (Berntorp *et al*., 2014). To recover an imperfect arrangement or post-process, the figured ideal setup reduces the intricacy of the express regulator.

Mathematical Formulation

It is challenging to acquire a mathematical model because of the convoluted wheel-ground kinematics limitations. A satisfactory mathematical model is achieved after making a few assumptions:

i. The vehicle's focal point coincides with its mathematical focus
ii. Vehicle motion pictures with the wheels consistently in contact on the ground
iii. Two wheels on either side turn at a similar speed while taking turns and slip is the only force acting on the tires (Gonzalez *et al*., 2009)

To design the optimal control by augmenting the system model with integral of the steady-state error. Here $f(X_k, U_k)$ is specified as a priority model and it will remain unchanged. The disturbance model $d(X_k, V_k, U_k)$ will compensate for the errors which the priority model does not minimize. The non-linear, state-space system X_{k+1} and the unknown V_{k+1} as *h*. Having U_k, as the control input at time *k*, separating states into two parts X_k, and Vk, and trying to formulate X_k through the path-tracking control model (Yu *et al*., 2015). Minimizing the cost function for the coming timesteps, the new Jacobian is determined:

$$X_{k+1} = f\left(X_k, U_k\right) + d\left(X_k, V_k, U_k\right) \tag{20}$$

$$V_{k+1} = h\left(V_k, U_k\right) \tag{21}$$

$$J(u) = \left[x_d - x\right]^T Q\left(x_d - x\right) + u^T Ru \tag{22}$$

State Reduction

In the model as continuous and, discrete-time systems are directly proportional to the time step. Carefully choosing continuous and discrete-time systems, as the time horizon and time step are affected. The time horizon and time step are also used to compute the number of nodes for the model. Another parameter of the cost function is to penalize the control inputs for getting a smooth control action. The cost function is minimized in each time step (Shladover *et al*., 1991):

$$X_r = \left[x_r \; y_r \; \theta_r \; \theta_b\right]^T \tag{23}$$

$$X_e = \begin{bmatrix} x_e \\ y_e \\ \alpha_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta_r & \sin\theta_r & 0 & 0 \\ -\sin\theta_r & \cos\theta_r & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_m - x_r \\ y_m - x_r \\ \theta_t - \theta_r \\ \theta_m - \theta_b \end{bmatrix} \tag{24}$$

where:

u_R = The forward velocity
$\dot{s}$ = The rate of progression
X_r = The state vector
X_e = The error state
$K_{r(s)}$ = The path curvature at the path lengths

The dynamic error model is chosen in a rotated coordinate frame and the prediction model includes an unmeasured disturbance. MPC expects to encounter disruptions in the plant and the UD model is an integrator with its input assumed to be white noise. The output from the model is added to the relative yaw angle. Although the calculations' presentation is sensibly acceptable, a few issues should be considered for the MPC-based yaw stability control as indicated by real Husky test or simulation results. The vehicle prediction formulation for vehicle yaw dependability control usually is around 0.2-0.4 sec (Dekkata *et al*., 2020):

$$\dot{x}_e = y_e k_r(s)\dot{s} - \dot{s} + u_R \cos\alpha_e \tag{25}$$

$$\dot{y}_e = -x_e k_r(s)\dot{s} + u_R \sin\alpha_e \tag{26}$$

$$\dot{\alpha}_e = \phi u_R - k_r(s)\dot{s} \tag{27}$$

$$\dot{\theta}_e = w_m - w_b \tag{28}$$

Constraints

The constraints add stability to the controller even if the prediction horizon is finite. The optimization of MPC is subjected to the mixed or general state of constraints; equality constraints are seldomly used in this research compared to inequality constraints. Using the equality constraints rarely because a non-linear equality constraint can never be satisfied in a finite number of algorithm iterations. MPC prediction step can have two constraints of the form (Richards and How, 2006):

Equality constraints:

$$k\big(x(t),u(t),t\big) = 0 \tag{29}$$

Inequality constraints:

$$h\big(x(t),u(t),\big) \geq 0 \tag{30}$$

Maximum velocity constraints, if all the inequality constraints are positive, which corresponds to the Lagrange multiples, an optimal solution for control problem is found. The line search boundary doesn' t devaluates the disparity requirements near the limit. The specific calculation boundaries size and diminishing pace rely upon the strategy for decision:

$$y_{\min} \leq y \leq y_{\max} \tag{31}$$

$$\Delta u_{\min} \leq y \leq \Delta u_{\max} \tag{32}$$

$$u_{\min} \leq y \leq u_{\max} \tag{33}$$

Generating the desired yaw rates to control the Husky using the bicycle model does not give back the tire's forces frictional limitations:

$$u \leq u_{\max} \tag{34}$$

Desired forward and rotational velocities:

$$u_d = \min\big(u_R\big) \tag{35}$$

Velocity magnitude constraints:

$$\sqrt{\frac{\mu g}{|k|}} \leq u \leq \sqrt{\frac{\mu g}{|k|}} \tag{36}$$

$$0 \leq u \leq \min\left(u_{\max}, u_d, \sqrt{\frac{\mu g}{|k|}}\right) \tag{37}$$

Husky's translation and rotation can be controlled separately because of omnidirectional; rewriting Eq. 38 as the decoupling translation and Eq. 39 as rotation:

$$X(t) = f\big(\dot{X}(t), U(t)\big),\ X(0) = X_0 \tag{38}$$

$$X(t) = \begin{bmatrix} x_m & y_m & \theta_m \end{bmatrix}^T \tag{39}$$

$$U(t) = \begin{bmatrix} u_m & v_m & w_m \end{bmatrix}^T \tag{40}$$

where:
$X(t)$ = The state vector
$U(t)$ = The vector of the 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 time the angular velocity gives us the tangential velocity of each wheel. Husky can move forwards or backward with only one axis: Using the kinematic model, velocity in a fixed structure is essential. After experiencing errors in the steady state in the yaw angle, minimized the lateral deviation to zero to eliminate the yaw angle's steady-state error. The plant model minimizes the steady-state error to control Husky's longitudinal and lateral motion (Wu *et al*., 2001).

Optimization criterion:

$$\begin{aligned} &= \sum_{k=0}^{N_p} \left\{ Q_1 \left[e_1(t_k) \right]^2 + Q_2 \left[e_2(t_k) \right]^2 + ... + Q_n \left[e_n(t_k) \right]^2 \right\} \\ &+ \sum_{k=1}^{N_c} \left\{ R_1 \left[\Delta u_1(t_k) \right]^2 + R_2 \left[\Delta u_2(t_k) \right]^2 + ... + R_r \left[\Delta u_r(t_k) \right]^2 \right\} \end{aligned} \tag{41}$$

Introducing two manipulate variables for longitudinal and lateral deviation and zero weights are applied in the plant model. The standard cost function and the state function are specified as Jacobian. From Eq. 41, the weighting matrices Q and R tune the relative importance of the output vector elements. For every state implementing a trajectory linearization control for Husky, reach the reference trajectory (Vougioukas, 2007), with minimum effect on the cross-track error.

Discrete or Continuous

While processed operate in constant time and do classical control laws such as PI, decision-making tends to be more of a discrete process. Decision-making requires processing time and cannot be instantaneous, especially interacting inputs/outputs/constraints and performance. Common predictive laws are implemented in discrete time.

Subject to:

$$0 = f\left(\frac{dx}{dt}, x, y, p\right) \tag{42}$$

$$0 \le g\left(\frac{dx}{dt}, x, y, p\right) \tag{43}$$

The model captures the time delays and plant interactions; the internal model predicts plant behavior over a future horizon in time. The optimizer and the controller adjust the control signals without violating the reference set points. Modeling errors are compensated by feedback and disturbance is rejected (Shojaei *et al*., 2009).

Transfer Function

The most common transfer function model with MPC is the so-called CARIMA model. This model subsumes in its structures many other popular forms. It is used because the uncertainty is included in a way that is a good presentation of slowly varying disturbances that could have a non-zero steady state:

$$\begin{array}{lll} x[1]=u[1] & x[5]=\dot{u}[2] & x[9]=\ddot{u}[3] \\ x[2]=\dot{u}[1] & x[6]=\ddot{u}[2] & x[10]=\phi \\ x[3]=\ddot{u}[1] & x[7]=u[3] & x[11]=\dot{\phi} \\ x[4]=u[2] & x[8]=\dot{u}[3] & x[12]=\ddot{\phi} \end{array} \tag{44}$$

With A and B matrices, the differentially flat system with controllable states is expressed as four independently translation states and sub-matrices A_s and B_s are defined as below:

$$A\left(\Delta t, a_r, a_\phi\right) = \begin{bmatrix} A_s\left(\Delta t, a_r\right) & 0 & 0 & 0 \\ 0 & A_s\left(\Delta t, a_r\right) & 0 & 0 \\ 0 & 0 & A_s\left(\Delta t, a_r\right) & 0 \\ 0 & 0 & 0 & A_s\left(\Delta t, a_\phi\right) \end{bmatrix} \tag{45}$$

$$B\left(\Delta_t, b_r, b_\phi\right) = \begin{bmatrix} B_s\left(\Delta t, b_r\right) \\ B_s\left(\Delta_t, b_r\right) \\ B_s\left(\Delta t, b_r\right) \\ B_s\left(\Delta t, b\phi\right) \end{bmatrix} \tag{46}$$

$$A_s\left(\Delta t, a\right) = \begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 1 & \Delta t \\ 0 & 0 & a \end{bmatrix} \tag{47}$$

$$A_s\left(\Delta t, a\right) = \begin{bmatrix} \frac{b\Delta t^2}{2} \\ b\Delta t \\ b \end{bmatrix} \tag{48}$$

In the system, the transfer function between the acceleration and the input has sample time Δt b_r, b_ϕ, a_r, a_ϕ as constants, the control order could be found exclusively with MPC. Nonetheless, because of the conceivably long reference trajectory and the tested states' critical importance, that alternative isn' t plausible (Wang and Qi, 2001). Because of the number of the autonomous boundaries of the quadratic development in the length of the horizon. The Husky direction would require the arrangement of the MPC with numerous stages of prediction stages. This is unreasonable, given the current innovation and the installed computational power of the Husky. On the other hand, the proposed method consistently addresses MPC with just a few inadequately conveyed steps because of not being obliged by Husky' s natural elements. The number of steps is a plain boundary, which relies upon the speed, given this can be tackled at 240 Hz progressively locally available with the robot (Fang *et al*., 2006):

$$a(z)y_k = b(z)u_k + T(z)\frac{\varsigma_k}{\Delta} \tag{49}$$

T(z) could arise from model identification; in practice, it is treated as design parameter with reference set points.

Concepts of Prediction

The predicted state variables are calculated along the predicted horizon using the future control input (Kraus *et al*., 2013):

$$\begin{aligned} &\hat{x}_p\left(k+N_p \mid k\right) = \tilde{A}x(k) + \tilde{B}U\left(k+N_c\right) \\ &+\tilde{E}W\left(k+N_p\right) + K_c\left(x(k) - \hat{x}_p\left(k \mid k-1\right)\right) \end{aligned} \tag{50}$$

$$\begin{aligned} &\hat{y}_p\left(k+N_p \mid k\right) = \tilde{C}x(k) + \tilde{D}U\left(k+N_c\right) \\ &+\tilde{G}W\left(k+N_p\right) - \tilde{F} + K_y\left(y(k) - \tilde{y}_p\left(k \mid k-1\right)\right) \end{aligned} \tag{51}$$

DOI: 10.3844/jmrsp.2022.90.105

The prediction model is based on the current measured state and the future control actions and the correction factor corrects the prediction. Discrete models are one step ahead in prediction models; given data at sample *k*, one can determine data at sample *k+1*:

$$a(z)y_k = b(z)u_k + d_k \tag{52}$$

$$a(z) = 1 + a_1 z^{-1} + \ldots\ldots + a_n z^{-n} \tag{53}$$

$$b(z) = b_1 z^{-1} + \ldots\ldots + b_m z^{-m} \tag{54}$$

$$y_{k+1} + a_1 y_k + \cdots + a_n y_{k-n+1} = b_1 u_k + b_2 u_{k-1} + \cdots + b_m u_{k-m+1} + d_k \tag{55}$$

$$y_{k+1} = b_1 u_k + b_2 u_{k-1} + \cdots + b_m u_{k-m+1} + d_k - a_1 y_k - \cdots - a_n y_{k-n+1} \tag{56}$$

The one-step-ahead prediction can be used recursively to find an n-step-ahead prediction as follows:

$$y_{k+1} + A_1 y_k + \cdots + A_n y_{k-n+1} = b_1 \Delta u_k + b_2 \Delta u_{k-1} + \cdots + b_m \Delta u_{k-m+1} \tag{57}$$

$$y_{k+2} + A_1 y_{k+1} + \cdots + A_n y_{k-n+2} = b_1 \Delta u_{k+1} + b_2 \Delta u_k + \cdots + b_m \Delta u_{k-m+2} \tag{58}$$

$$y_{k+3} + A_1 y_{k+2} + \cdots + A_n y_{k-n+3} = b_1 \Delta u_{k+2} + b_2 \Delta u_{k+1} + \cdots + b_m \Delta u_{k-m+3} \tag{59}$$

$$y_{k+4} + A_1 y_{k+3} + \cdots + A_n y_{k-n+4} = b_1 \Delta u_{k+3} + b_2 \Delta u_{k+2} + \cdots + b_m \Delta u_{k-m+4} \tag{60}$$

Using the 1 step ahead to find y_{k+1}, then substitute this into the following Eq. 57 to find y_{k+2}, then use y_{k+1} and y_{k+2} to determine y_{k+3} and keep the iterating through y_{k+n}. Solving for the outer predictions, simultaneous equations can be represented using matrix/vector format, after which the solution is obvious. An illustration and some suitable notations are given here, which separates the past and future variables:

$$y_{k+1} + A_1 y_k + \cdots + A_n y_{k-n+1} = C_A \left[y_{k+1}\, y_{k+2}\, y_{k+3}\, y_{k+4} \right] + H_A \left[y_k\, y_{k-1}\, y_{k-2}\, y_{k-3} \right] \tag{61}$$

GPS Measurements

Using GPS coordinates to localize the Husky in the environment. Based on the high-level decisions and generated path (whether to go straight or turn), the following immediate waypoints will be generated. Then, GPS/IMU sensor readings will then be used to control the Husky to follow the generated waypoints by employing the controller. After initial tests to determine the GPS/IMU system' s accuracy nearly 5-8 cm by placing markers collecting various data. Where GPS signals are available in outdoor environments, position and velocity can be integrated from IMU at the rate of 1 Hz; in the loosely coupled the measurement model for GPS/IMU as:

$$\left[p_{gps} - p_{imu} \;\; v_{pgs} - v_{imu} \right] = \left[\delta_p \, \delta_v \right] \tag{62}$$

Navigation

Husky navigates through a series of waypoints/points of interest to perform autonomous navigation by avoiding obstacles and bring the Husky to the destination point. Leveraging the navigation ArcMap to search and geocode points of interest by utilizing ArcMap functionalities. Based on Husky' s current location of the point of interest and the destination address location, determining the shortest route by parsing the map data.

Routing

The map data comes in Protobuf format and offset encoded to minimize their storage requirements. The maps' logical data model is based on the navigation map that describes the surface topology based on the links, nodes, polylines, shape points, and their attributes. Creating a switching weighted directional graph using links and nodes to transform the routing problem into finding the maps' shortest path. The weight of an edge in the graph is the length of a link between two nodes. It is the length of the path along the segment. Adopting the map to be a switching graph to handle static obstacles and simplify the rerouting problem.

Figure 6 the Ark bridge map is illustrated, the Ark bridge map shows the set of waypoints as the desired trajectory. After creating a switching map, applying Dijkstra' s algorithm to find the shortest path. In this case, assuming the minimum number of dynamic obstacles on the route to the destination. The approach for routing initially planned to import the map data and geocode the points of interest. Husky' s current location is sent to the controller and from the map data, the nearest node to Husky' s location is discovered. Set of links that connect the node to the first node and form the shortest path. Then, proceed to find a set of connections that create the most straight forward route among consecutive nodes. Return set of ordered links from Husky' s current location to the destination.

Rerouting

The Husky may face dynamic and static obstacles from the current location to the destination address. In such scenarios, the Husky is supposed to reroute to avoid obstacles and navigate through obstacles free path.

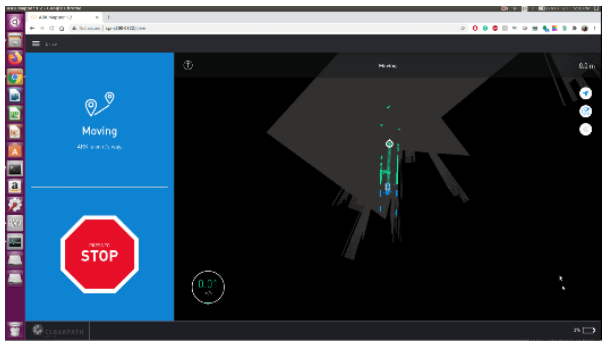

Fig. 6: Ark bridge map

Using lidar data and vision-based detection to determine the position of obstacles along the path (Muktadir and Yi, 2021). In the lidar detection algorithm, considering the obstacles that Husky might encounter during navigation. In vision-based detection, training the convolution neural network to detect the obstacles blocking the path (Garfo *et al*., 2020). Once noticing the obstacles on the way, localizing the location of the obstacles by finding the nearest node to the obstacle by utilizing the map data. Localizing other objects helps us identifying the link on which the obstacle is located. Updating the weighted directional graph by removing the edge representing the obstacles on the path, Husky can reroute. After updating the map, applying Dijkstra' s algorithm to find the shortest route. The navigation framework has three essential parts: localizations, path planning, and motion control (Tereda, 2021).

Figure 7 EKF algorithm is picturized in a block diagram, from setting the initial values in the first step to the final step until the compute error is minimized and covariance is adjusted. The localization has three significant steps, in the first step, the GPS and IMU data are fused using an Extended Kalman Filter (EKF). The shortest route is generated by Dijkstra's algorithm, which is characterized by a vector of nodes and node ID, a vector of links, and a link ID beside the map tile ID. Based on these, it is extracting the position of the Husky in *x and y* coordinates. Husky should localize itself and other objects on the path to precisely navigate along the route. It should determine a drivable path and predict the behavior of different obstacles that are moving around Husky. This issue is addressed in the navigation framework below.

Localization

Localization helps the Husky determine its position on the map. Finding the Husky' s location on the map requires the active sensor data from fusing all the sensors data with GPS and IMU data.

Furthermore, localization helps Husky determine other objects' s location on the path and thereby predict their behavior. Set the initial values and predict the state and error covariance as shown in Eq. 63, 64:

$$\hat{x}_{\bar{k}} = f\left(\hat{x}_{\bar{k}-1}\right) \tag{63}$$

$$p_{\bar{k}} = AP_{k-1}A^T + Q \tag{64}$$

Localization helps Husky determine other objects' location on the path and compute the Kalman gain from Eq. 65 to fuse the GPS and IMU data using an Extended Kalman Filter (EKF). Computing the state estimates from the measurements in Eq. 66, the system' s final assessment is calculated. Finally, compute the error covariance in Eq. 67 and send it to the predicted state to reduce the error:

$$K_k = P_{\bar{k}}H^T\left(HP_{\bar{k}}H^T + R\right)^{-1} \tag{65}$$

$$\hat{x}_k = \hat{x}_{\bar{k}} + K\left(z_k - h\left(\hat{x}_{\bar{k}}\right)\right) \tag{66}$$

$$P_k = P_{\bar{k}} - KHP_{\bar{k}} \tag{67}$$

Set localization estimates the pose, linear velocity, and yaw rate of Husky in real-time is the basis for Husky navigation. Using the map data with the combination of GPS and IMU for real-time, in other words, leveraging the GPS and IMU sensors data fusion to interface with the accurate positioning. In this research, the localization method fuses the GPS and IMU data using an EKF and determines the detected obstacles' lateral distance (Razvan and Rafaila, 2015). The localization has three significant steps: In the first step, the GPS and IMU data are fused using an EKF (Hoshiya and Saito, 1984). The second step involves determining the Husky position in bird's eye view in *x*, and *y* coordinates from the map data. In the final stage, the map matches the surrounding information from sensors with the available map data. The localization algorithm will match the detected obstacles to the map coordinate from the map data. The estimation of the Husky state using EKF is based on the following kinematic model of the Husky:

$$\dot{x} = v\cos\left(\theta + \beta\right) \tag{68}$$

$$\dot{y} = v\sin\left(\theta + \beta\right) \tag{69}$$

$$\dot{\theta} = \frac{v}{L_f}\delta \tag{70}$$

$$\dot{v} = a \tag{71}$$

where:
v = The velocity of the Husky
a = The acceleration of the Husky

From the above Eq. 68 to 71 having v is the velocity of the Husky, a is the acceleration, and x, y are coordinates of the center of the mass of the Husky in the frame. The steering angle and the acceleration are input to the system. Let the kinematic and the measurement model be given as:

$$\dot{x} = f\left(x\right) + w \tag{72}$$

$$z = h\left(x\right) + v \tag{73}$$

where:
x = The state of the Husky
w = The process noise
v = The measurement noise

Assuming *p(w)~N (0, Q)* and *p(v)~N (0, R)* and determining Jacobean matrices as:

$$A \equiv \frac{df}{dx}\Big|\bar{x}_k \tag{74}$$

$$H \equiv \frac{dh}{dx}\Big|\bar{x}_k \tag{75}$$

The IMU measurement data is used in the prediction stage, while the GPS's measurement data is used to estimate the Husky pose from the GPS. After fusing GPS and IMU, the next step is determining the detected obstacles' lateral distance and drivable path from the map data. The drivable path is modeled using clothoid in the camera reference frame, with a third-order Taylor expansion can approximate. The slope, curvature, and curvature derivative of the drivable path is calculated and marked. The lateral distance from the drivable path is calculated and minimized by reducing the cross-track error. It is used in conjunction with map data to determine the current position of the Husky.

Localizing other objects after Husky determines its position, it classifies other objects around it, and relies on lidar and camera detections to localize other objects on the map. Husky assigns a particular link to each object based on the object' s location on the map coordinate system. The objects around Husky can be static, dynamic and the dynamic obstacles may vary in speed. Assuming detected objects' rate doesn' t considerably differs, implementing a Kalman filter algorithm with a single integrator equation of motion to predict their behavior. For solving the optimization problem, the control signals in system matrix evaluated with description of the system:

$$\min_{v_{[t]}, x_{[t]}} U(x,v) = \frac{1}{2}\sum_{i=1}^{m-1}\left(e_{[i]}^T \mathrm{Re}_{[i]} + v_{[i]}^T S v_{[i]}\right) \tag{76}$$

$$x_{[t+1]} = Ax_t + Bv_t \tag{77}$$

$$x_{[t]} \le x\_\max_{[t]} \tag{78}$$

$$x_{[t]} \ge x\_\min_{[t]} \tag{79}$$

The control error's notion with the horizon is predicted at the input control action (Mayne *et al*., 1988). Penalizing the error using the cost function from the quadratic semi-positive definite constraints. At a predefined zone, probability capacities are made to determine the probability of restricting the item at a given area. The probability capacities are converged into more extensive planning, which keeps communicating the objects' undoubted location.

Husky's control algorithm for detecting the Most Important Object (MIO) of numerous detections from the lidar sensor are produced from the Probability Density Function (PDF) format to amplify the probability of recognizing the object interest. This methodology' s principal potential constraint is that it may not effectively reach out to a self-assertive number of items. Limiting a moving object detection would be a complicated process (Frasch *et al*., 2013).

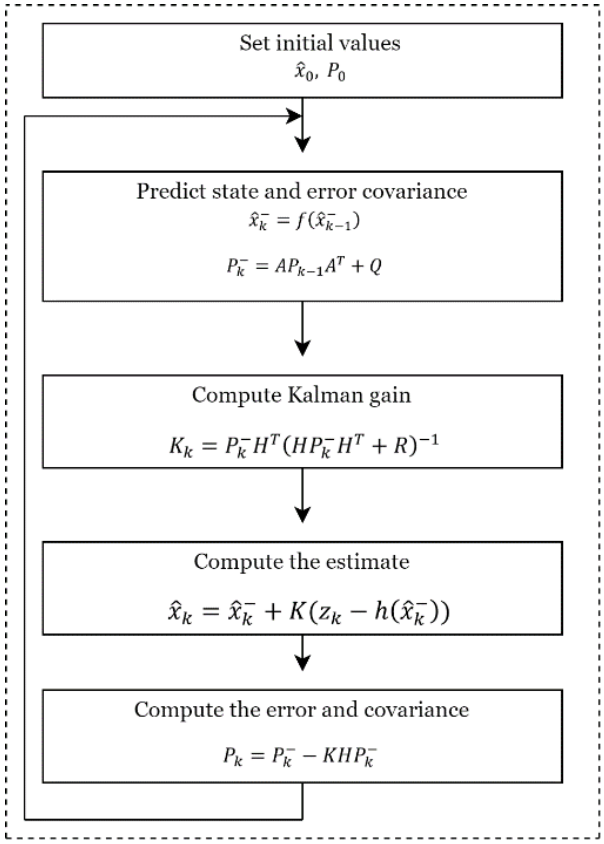

Fig. 7: EKF algorithm

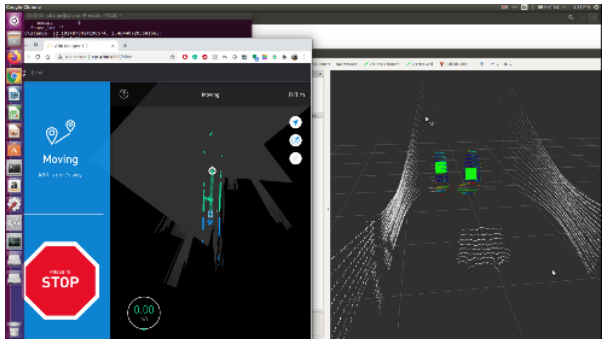

Fig. 8: Localizing other objects

Figure 8 detecting multiple objects with respect to Husky relative motion, LiDAR detection helps in detecting objects around the Husky and localize the objects in the map. The detected objects are tracked to avoid collision with the Husky using Kalman filter algorithm.

Path Planning

The main objective of path planning is to extract waypoints from the start to destination address point, followed by generating a feasible and smooth trajectory. The waypoints are segmented into several center coordinate points between the links extracted from the map. Path planning relies on the information from localization, perception, tracking, and map data to navigate Husky and a series of addresses and point of interests.

In the behavior planner, implementing heuristic conditions based on the Husky' s coordinate location, the path in which the Husky is moving, location of objects, their respective positions in the map are evaluated and distance is calculated from the behavior point of interest of the Husky. Then, for each action, determining a safe target speed that respects the Husky speed limitations. Relying on map data and sensor detection data to generate suitable waypoints from start to destination while executing the missions safely. Geocode the point of interest and destination address from a weighted graph representing the map data' s network. Implementing Dijkstra' s shortest path algorithm to find the shortest route from start to the destination address, localizing the Husky considering the objects detected by sensors, and determining the Husky behavior considering the action of the objects around the Husky.

Figure 9 the architecture design of MPC with mission planner, the input commands from mission planner helps the Husky movement through a feed forward system. As MPC generates the difference between the actual states of the Husky to reference, the controller' s predicted output is sent to feed-forward action to calculate the error-free results, which will be applied to the real-time system. From the reference trajectories, defining the control inputs for the feed-forward system. The reference control velocity for the Husky and the yaw rate for the Husky are calculated from the reference trajectory:

$$v_r = \pm\sqrt{\left(\dot{x}_r^h\right)^2 + \left(\dot{y}_r^h\right)^2} \tag{80}$$

$$\gamma_r^h = \frac{\dot{x}_r^h \ddot{y}_r^h - \dot{y}_r^h \ddot{x}_r^h}{\left(\dot{x}_r^h\right)^2 + \left(\dot{y}_r^h\right)^2} \tag{81}$$

The positive sign indicates that Husky moves forward and the negative sign suggests that Husky navigates in the reverse direction. Control action for feed-forward is calculated for Husky, assuming steering angle is small and considering the steady-state behavior between steering angle and longitudinal velocity is considered:

$$\delta_r^h = \frac{\gamma_r^h L^h}{vr} \tag{82}$$

In nature, trajectory generation is considered a non-linear problem. This study uses the non-linear feedback controller, so Husky can be asymptotically stabilized. The internal reference frame is attached to the ground because they are considered moving frames. The error state expressed in the Husky frame is written below:

$$Y_e = T * \left[Y_r - Y\right] \tag{83}$$

Between the reference frames, T is the transformation matrix:

$$T = \begin{bmatrix} \cos\left(\varphi^t\right) & \sin\left(\varphi^t\right) & 0 & 0 \\ -\sin\left(\varphi^t\right) & \cos\left(\varphi^t\right) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{84}$$

Calculating the error state and deriving the system model into the below Eq. 85, 86, and 87:

$$\dot{x}_e^h = \gamma^h y_e^h - v + v_r \cos\left(\varphi_e^h\right) \tag{85}$$

$$\dot{y}_e^h = -\gamma^h x_e^h + v_r \sin\left(\varphi_e^h\right) \tag{86}$$

$$\dot{\varphi}_e^h = \frac{v_r * \tan\left(\delta_r^h\right) - v * \tan\left(\delta^h\right)}{L^h} \tag{87}$$

Linearizing the error model, the Husky trajectory in the state space with the reference trajectory is below:

$$\dot{Y}_e = AY_e + Bu_e \tag{88}$$

$$\dot{Y}_e = \begin{bmatrix} 0 & \gamma_r^h & 0 & 1 \\ -\gamma_r^h & 0 & 0 & 0 \\ 0 & 0 & v_r & 0 \\ 0 & 0 & 0 & -\frac{1}{\tau} \end{bmatrix} Y_e + \begin{bmatrix} 0 & 0 & 0 \\ \frac{v_r}{L_h} & 0 & 0 \\ \frac{v_r L_d}{L_h} & 0 & 0 \\ 0 & 0 & \frac{K}{\tau} \end{bmatrix} u_e \tag{89}$$

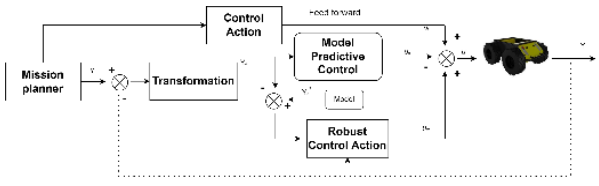

Fig. 9: Block diagram for feedback MPC

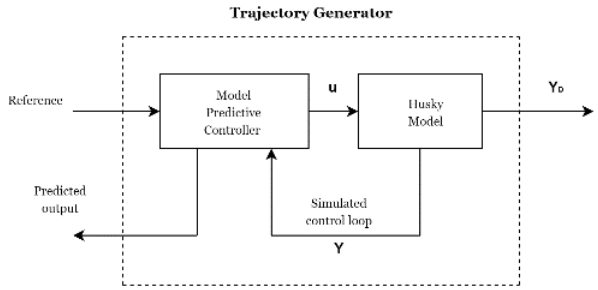

Fig. 10: Trajectory generator block diagram

The control vectors and state vectors for error-based models on the trajectory tracking are Y_e and u_e (Klančar and Škrjanc, 2007):

$$Y_e = \left[x_e^h \; y_e^h \; \varphi_e^h \; v_e \right]^T \tag{90}$$

$$u_e = \left[\delta_e^h \;\; \lambda_e \right]^T \tag{91}$$

Figure 10 the desired trajectory is generated from the lateral controller for Husky, and the main purpose of the lateral controller is to adjust the Husky heading angle to the desired path. A smooth trajectory is generated from the extracted waypoints for a smooth movement by fitting a third-order polynomial function. For this purpose, it is important to transform the waypoints coordinates from the global frame to Husky' s frame, then select the first six waypoints coordinates to fit the cubic polynomial function. This helps the system to determine the y-coordinate and the heading direction of the Husky:

$$y = f_m(x) = a_3 x^3 + a_2 x^3 + a_2 x^2 + a_1 x + a_0 \tag{92}$$

$$\tan\theta = \frac{dy}{dx} 3a_3 x^2 + 2a_2 x + a_1 \tag{93}$$

Results

Control Design Implementation

After generating a smooth trajectory, designing the MPC to control the Husky' s longitudinal and lateral motion at each sampling timestep, an optional control problem is solved over a finite horizon. The optimal control signal is applied to the Husky during the following sample interval. A new optimal control problem based on unique measurements is solved over a shift horizon at the next time step. Therefore, finding the optimal solution relies on the Husky model, input and output constraints, and the cost function adopted to capture the trajectory tracking problem. Actuations are forward and backward torque and lateral motion of the Husky, applying the prediction to the models' state at sample time *t+1*, considering the previous state as *t* (Quirynen *et al*., 2012):

$$x_{t+1} = x_t + v * \cos(\varphi) * dt \tag{94}$$

$$y_{t+1} = y_t + v * \sin(\varphi) * dt \tag{95}$$

$$\varphi_{t+1} = \varphi_t + \delta_t \frac{v_t}{L_f} * dt \tag{96}$$

$$v_{t+1} = v_t + a_t * dt \tag{97}$$

The distance between the Husky's center of gravity and the front center of the tires is L_f, vector $[x\ y\ \varphi\ v]$ as the Husky state and actuation $[\delta\ a]$.

Input Trajectory

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

Errors and Cost Function

Defining new states to capture the transformed model from a global frame to a mobile frame. These errors are cross-track error in Eq. 98 and orientation error in Eq. 99:

$$cte_{t+1} = f_m(x_t) - y_t + v * \sin(\varphi) * dt \tag{98}$$

$$e\varphi_{t+1} = \delta \frac{v_t}{L_f} * dt \tag{99}$$

Cross-track error is the difference between the current position of the Husky and the desired position:

$$cte_t = y_t - y_{desired} \tag{100}$$

For the input trajectory, computing the desired values of *y* by fitting a polynomial and computing for *x* at time *t*:

$$y_{desired} = f(x_t) = ax^3 + bx^2 + cx + d \tag{101}$$

Cross-track error is computed accordingly at time *t+1*, considering error at *t* and error at the next step:

$$\begin{aligned} cte_{t+1} &= cte_t + v_t * \sin(e\varphi) dt \\ &= y_t - f(x) + v_t * \sin e(\varphi) * dt \end{aligned} \tag{102}$$

Orientation error is the difference between the current orientation and desired orientation of the Husky:

DOI: 10.3844/jmrsp.2022.90.105

$$e\varphi_t = \varphi_t - \varphi_{desired} \quad (103)$$

For the input trajectory computing the desired values of φ by fitting a polynomial and computing for x at time t:

$$\varphi_{desired} = \arctan\left(f'\left(x_t\right)\right) = \arctan\left(3ax^2 + 2bx^1 + c\right) \quad (104)$$

The orientation error is computed at time $t+1$, considering the error at t and error at the next step:

$$e\varphi_{t+1} = e\varphi_t + \frac{v_t}{L_f} * \delta_t \, dt = \varphi_t - \arctan\left(f'\left(x_t\right)\right) + \frac{v_t}{L_f} * \delta_t * dt \quad (105)$$

Building the cost function considering the cross-track error and orientation error to guide the vehicle near the reference trajectory. To keep the Husky moving, the δ, which is the difference between the current speed and the reference speed. Setting constraints on maximum longitudinal and lateral range for the Husky.

Implementation

During Implementation, adding weights at cost function to each term to identify each term' s impact on the streamlining the error. Cross-track error weight is for stability, adding higher weights to make sure Husky is close to the desired trajectory. From Fig. 11, the mean error is -0.1.

Figure 11 higher the weight, lower the cross-track error. Cross-track error is reduced when higher weights are applied. As applying higher weights, the Husky tends to stay close to the desired trajectory.

Figure 12 lower the weight, higher the cross-track error. As applying lower weight, the Husky tends to move away from the desired trajectory. The weighting matrices Q and R tune the relative importance of the output vector elements. For every state implementing a trajectory linearization control for Husky and the Husky reaches the reference trajectory with minimum effort on the cross-track error. From Fig. 12, the mean error is -0.3 Orientation error is for stability, adding higher weights to make sure Husky is close to the desired orientation. From Fig. 13, the mean error is 0.02.

Figure 13 higher the weight, lower the orientation error. Orientation error is reduced when higher weights are applied.

Figure 14 lower the weight, higher the orientation error. Applying lower weight, Husky tends to oscillate or move abruptly away from the center.

Figure 15 the higher the actuation weights, Husky tends to stay close to the reference path. Higher the actuation weight, the smoother the trajectory. The lateral motion of the Husky is smooth with the higher actuation weights. Actuation weights, higher weights on the actuation are used to reduce the actuation error because higher actuation smoother the trajectory.

Figure 16 a lower actuation weight tends the Husky to swirl away from the center. Lower actuation weight, uneven the trajectory. Applying lower actuation weight, more jerky the response. Comparing the higher actuation weights and lower actuation weights in Fig. 15 and 16.

Actuation delta weights, adding a delta between the subsequent timesteps and actuation also reduces the jerkiness during the movement of Husky. Comparing the higher actuation delta weight and lower actuation delta weights in Fig. 17 and 18.

Figure 17 the higher the actuation delta weight, smoother the trajectory. The longitudinal motion of the Husky is much smoother when the actuation delta weights are increased.

Figure 18 lower the actuation delta weights, uneven the trajectory. More jerky response for the lower actuation delta weights.

Fig. 11: Higher the weights, the lower the cross-track error

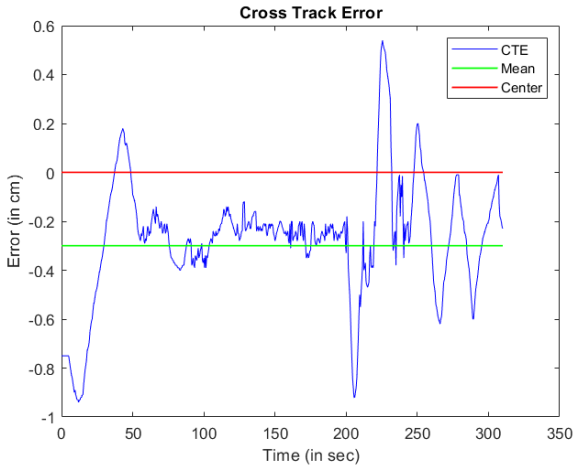

Fig. 12: Lower the weights, the higher the cross-track error

DOI: 10.3844/jmrsp.2022.90.105

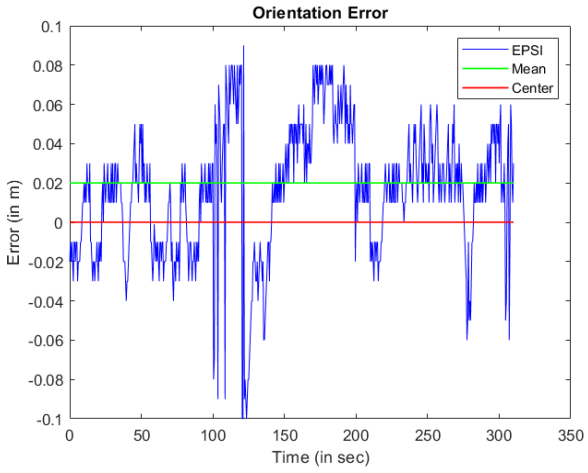

Fig. 13: Higher the weights, the lower the orientation error

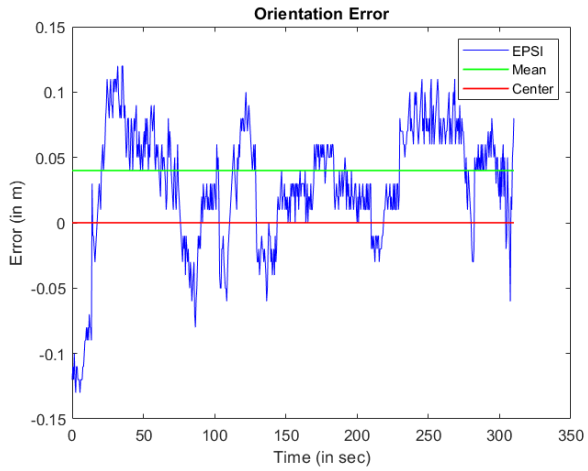

Fig. 14: Lower the weights, the higher the orientation error

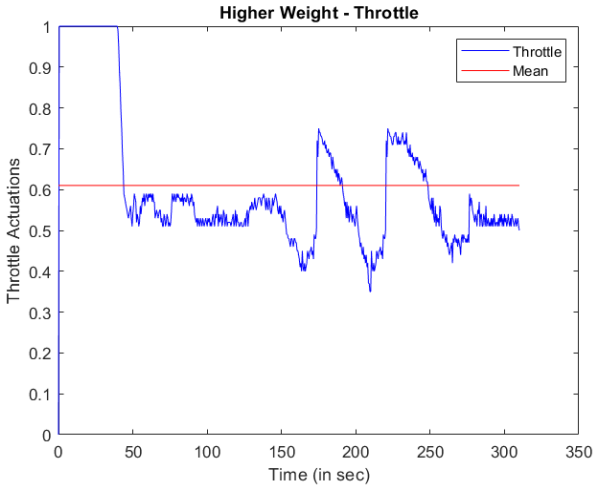

Fig. 15: Higher actuation weights

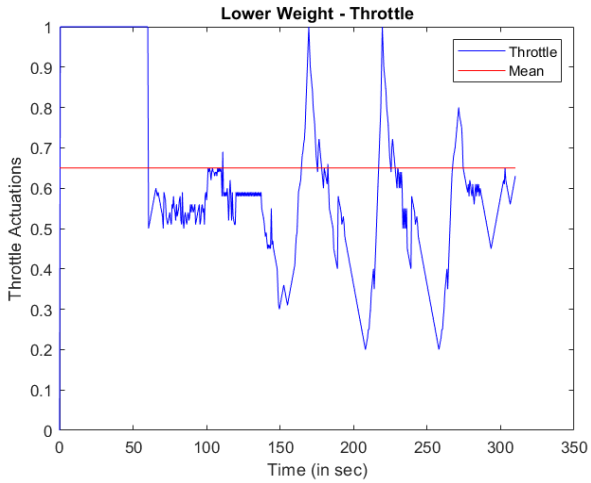

Fig. 16: Lower actuation weights

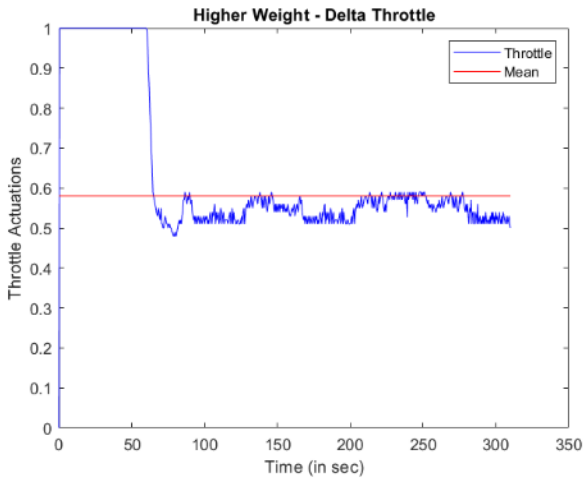

Fig. 17: Higher actuation delta weights

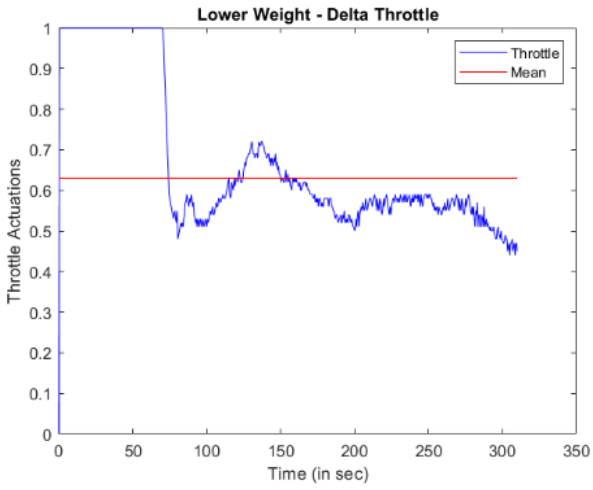

Fig. 18: Lower actuation delta weights

Conclusion

The Husky A200 ground robot can navigate autonomously with waypoints, and it is tested indoors in unstructured conditions without reduced data about the obstructions. The significance of this research is to present a new obstacle avoidance algorithm calculation MPC based, which improves longitudinal speed and guiding point while exploring the environment securely and reaching the target area as quickly as possible. Even with the limitations, the acceleration of the Husky showed much greater results in experimental runs. The system was tested using the data sequences (like image frames and laser points) with poses and lightning conditions. The Lidar helps to track the objects around the Husky and gives the distances from the obstacles. The MPC time length prediction horizon fluctuates because a variable speed direction is arranged until the sensor range. The Husky's power calculation from motor and brake elements is thought through for Husky's longitudinal sped limits to reduce the jerk. The controller depends on the active sensor data such as static and dynamic objects; based on the sensor data, a feasible path is generated. By reducing the cross-track error and orientation error in an active set method the MPC is optimized. The best possible route is extracted, and trajectory is generated based on Husky's current events and position. Husky can avoid obstacles for optimal path planning and target following. To provide a smooth trajectory for the Husky, cost function formulation is assigned a more significant value to reduce the more prominent variations. Multiple arrangements of mathematical creations are directed to exhibit the adequacy of the calculation. The kinematic and dynamic model of the robot presented in this research is used for transient and steady-state characteristics. The waypoints improve the execution of the Husky by permitting it to navigate to the dynamical limits. In future, this research work will require more vehicle models which consider variable value describing the slope in the most linear region. This helps MPC adjust the robot's lateral, longitudinal, yaw motions and helps approximate a continuous trajectory with discrete path to command behaviors by improving the kinematic and dynamic models of the robot.

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 Center for Advanced Transport Mobility and 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 for the unmanned ground vehicle.

Sun Yi: Reviewed the journal and supervised the research.

MA Muktadir: Tuned lidar detection algorithm to determine the position of obstacles.

Selorm Garfo: Trained the convolution neural network to detect the obstacles

Xingguang Li: Worked on kinematic model to simplify the Husky physics.

Amanuel Abrdo Tereda: Helped in conducting the experiments.

Ethics

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

References

Bahadorian, M., Savkovic, B., Eaton, R., & Hesketh, T. (2011). Toward a robust model predictive controller applied to mobile vehicle trajectory tracking control. *IFAC Proceedings Volumes*, *44*(1), 13552-13557. https://doi.org/10.3182/20110828-6-IT-1002.01786

Berntorp, K., Olofsson, B., Lundahl, K., & Nielsen, L. (2014). Models and methodology for optimal trajectory generation in the safety-critical road–vehicle manoeuvres. *Vehicle System Dynamics*, *52*(10), 1304-1332. https://doi.org/10.1080/00423114.2014.939094

Carlos, E., & Garcia, D. M. (1989). Model Predictive Control: Theory and practice -A survey. Automatica, 335-348. https://doi.org/10.1016/0005-1098(89)90002-2

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). https://www.proquest.com/openview/3b32319f9b4b89adf8778167b9c848c8/1?pq-origsite=gscholar&cbl=18750

Dekkata, S. C. (2021). *Model Predictive Control for Unmanned Ground Vehicles Using Robot Operating System* (Doctoral dissertation, North Carolina Agricultural and Technical State University). https://www.proquest.com/openview/3f97b9c10e4afc7b97d3cedf7cbe869b/1?pq-origsite=gscholar&cbl=18750&diss=y

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://pdfs.semanticscholar.org/b3ca/d00bed76e379a3a5690a92e86e3d24b793c7.pdf

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

Fang, H., Fan, R., Thuilot, B., & Martinet, P. (2006). Trajectory tracking control of farm vehicles in presence of sliding. *Robotics and Autonomous Systems*, *54*(10), 828-839. https://doi.org/10.1016/j.robot.2006.04.011

Ferreau, H. J., Bock, H. G., & Diehl, M. (2008). An online active set strategy to overcome the limitations of explicit MPC. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, *18*(8), 816-830. https://doi.org/10.1002/rnc.1251

Frasch, J. V., Gray, A., Zanon, M., Ferreau, H. J., Sager, S., Borrelli, F., & Diehl, M. (2013, July). An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles. In *2013 European Control Conference (ECC)* (pp. 4136-4141). IEEE. https://doi.org/10.23919/ECC.2013.6669836

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

Gonzalez, R., Fiacchini, M., Guzmán, J. L., & Alamo, T. (2009, December). Robust tube-based MPC for constrained mobile robots under slip conditions. In *Proceedings of the 48 h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference* (pp. 5985-5990). IEEE. https://doi.org/10.1109/CDC.2009.5400508

Hoshiya, M., & Saito, E. (1984). Structural identification by extended Kalman filter. *Jour. of Eng. Mech., ASCE*, *110*(12).

Kiencke, U., & Nielsen, L. (2005). Vehicle modelling. *Automotive Control Systems: For Engine, Driveline and Vehicle*, 301-349. https://doi.org/10.1007/b137654

Klančar, G., & Škrjanc, I. (2007). Tracking-error model-based predictive control for mobile robots in real time. *Robotics and Autonomous Systems*, *55*(6), 460-469. https://doi.org/10.1016/j.robot.2007.01.002

Ko, Y. E., & Song, C. K. (2010). Vehicle modeling with nonlinear tires for vehicle stability analysis. *International Journal of Automotive Technology*, *11*(3), 339-344. https://link.springer.com/article/10.1007/s12239-010-0042-0

Kraus, T., Ferreau, H. J., Kayacan, E., Ramon, H., De Baerdemaeker, J., Diehl, M., & Saeys, W. (2013). Moving horizon estimation and nonlinear model predictive control for autonomous agricultural vehicles. *Computers and Electronics in Agriculture*, *98*, 25-33. https://doi.org/10.1016/j.compag.2013.06.009

Kuhne, F., Lages, W. F., & da Silva Jr, J. G. (2004, September). Model predictive control of a mobile robot using linearization. In *Proceedings of mechatronics and robotics* (Vol. 4, No. 4, pp. 525-530). https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.461.2651&rep=rep1&type=pdf

Mayne, D. Q., & Michalska, H. M. (1988). Receding horizon control of nonlinear systems. Proceedings of the 27th IEEE Conference on Decision and Control. Austin, TX: IEEE. https://doi.org/10.1109/CDC.1988.194354

Mayne, D. Q., Raković, S. V., Findeisen, R., & Allgöwer, F. (2009). Robust output feedback model predictive control of constrained linear systems: Time varying case. *Automatica*, *45*(9), 2082-2087. https://doi.org/10.1016/j.automatica.2009.05.009

Maynea, D. Q., Seronb, M. M., & Rakovića, S. V. (2005). Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, 219-224. https://doi.org/10.1016/j.automatica.2004.08.019

Mikael Eklund, J., Sprinkle, J., & Shankar Sastry, S. (2012). Switched and Symmetric Pursuit/Evasion Games Using Online Model Predictive Control with Application to Autonomous Aircraft. *IEEE Transactions on Control Systems Technology*, *20*(3), 604-620. https://doi.org/10.1109/TCST.2011.2136435

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

Quirynen, R., Vukov, M., & Diehl, M. (2012). Auto generation of implicit integrators for embedded NMPC with microsecond sampling times. *IFAC Proceedings Volumes*, *45*(17), 175-180. https://doi.org/10.3182/20120823-5-NL-3013.00013

Razvan, C., & Rafaila, G. L. (2015). Nonlinear model predictive control for autonomous vehicle steering. International Conference on System Theory, Control and Computing. Cheile Gradistei, Romania: IEEE. https://doi.org/10.1109/ICSTCC.2015.7321337

Richards, A., & How, J. (2006, June). Robust stable model predictive control with constraint tightening. In *2006 American Control Conference* (pp. 6-pp). IEEE. https://doi.org/10.1109/ACC.2006.1656440

Shladover, S. E., Desoer, C. A., Hedrick, J. K., Tomizuka, M., Walrand, J., Zhang, W. B., ... & McKeown, N. (1991). Automated vehicle control developments in the PATH program. *IEEE Transactions on Vehicular Technology*, *40*(1), 114-130. https://doi.org/10.1109/25.69979

Shojaei, K., Tarakameh, A., & Shahri, A. M. (2009, August). Adaptive trajectory tracking of WMRs based on feedback linearization technique. In *2009 International Conference on Mechatronics and Automation* (pp. 729-734). IEEE. https://doi.org/10.1109/ICMA.2009.5246126

Tereda, A. A. (2021). *Path Planning and Sensing for Autonomous Control of Robot Manipulators* (Doctoral dissertation, North Carolina Agricultural and Technical State University). https://www.proquest.com/docview/2622624694?pq-origsite=gscholar&fromopenview=true

Vougioukas, S. G. (2007, April). Reactive trajectory tracking for mobile robots based on nonlinear model predictive control. In *Proceedings 2007 IEEE International Conference on Robotics and Automation* (pp. 3074-3079). IEEE. https://doi.org/10.1109/ROBOT.2007.363939

Wang, D., & Qi, F. (2001, May). Trajectory planning for a four-wheel-steering vehicle. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)* (Vol. 4, pp. 3320-3325). IEEE. https://doi.org/10.1109/ROBOT.2001.933130

Wu, D., Zhang, Q., & Reid, J. F. (2001). Adaptive steering controller using a Kalman estimator for wheel-type agricultural tractors. *Robotica*, *19*(5), 527-533. https://doi.org/10.1017/S0263574701003459

Xie, F., & Fierro, R. (2008, June). First-state contractive model predictive control of nonholonomic mobile robots. In *2008 American control conference* (pp. 3494-3499). IEEE. https://doi.org/10.1109/ACC.2008.4587034

Yoon, Y., Shin, J., Kim, H. J., Park, Y., & Sastry, S. (2009). Model-predictive active steering and obstacle avoidance for autonomous ground vehicles. *Control Engineering Practice*, *17*(7), 741-750. https://doi.org/10.1016/j.conengprac.2008.12.001

Yu, S., Li, X., Chen, H., & Allgöwer, F. (2015). Nonlinear model predictive control for path following problems. *International Journal of Robust and Nonlinear Control*, *25*(8), 1168-1182. https://doi.org/10.1002/rnc.3133

Zanon, M., Frasch, J. V., & Diehl, M. (2013, July). Nonlinear moving horizon estimation for combined state and friction coefficient estimation in autonomous driving. In *2013 European Control Conference (ECC)* (pp. 4130-4135). IEEE. https://doi.org/10.23919/ECC.2013.6669832