2017 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
AUTONOMOUS GROUND SYSTEMS (AGS) TECHNICAL SESSION
AUGUST 8-10, 2017 - NOVI, MICHIGAN

ROBUST VEHICLE STABILITY BASED ON NON-LINEAR MODEL PREDICTIVE CONTROL AND ENVIRONMENTAL CHARACTERIZATION

Velislav Stamenov
Department of
Mechanical Engineering
Auburn University
Auburn, AL

Stephen Geiger
Department of
Mechanical Engineering
Auburn University
Auburn, AL

David Bevly, Ph.D.
Department of
Mechanical Engineering
Auburn, AL

Cristian Balas
TARDEC
Warren, MI

ABSTRACT

A Non-linear Model Predictive Controller (NMPC) was developed for an unmanned ground vehicle (UGV). The NMPC uses a particle swarm pattern search algorithm to optimize the control input, which contains a desired steer angle and a desired longitudinal velocity. The NMPC is designed to approach a target whilst avoiding obstacles that are detected using a light detection and ranging sensor (lidar). Since not all obstacles are stationary, an obstacle tracking algorithm is employed to track obstacles. Two point cluster detection algorithms were reviewed, and a constant velocity Kalman filter-based tracking loop was developed. The tracked obstacles' positions are predicted using a constant velocity model in the NMPC; this allows for avoidance of both stationary and dynamic obstacles.

INTRODUCTION

Unmanned ground vehicles (UGV) are very versatile. They may be used for transportation of items and/or persons in situations where a driver is not available or in situations where the area of travel is too dangerous for humans. UGVs also allow all passengers to focus on other more pressing tasks during travel.

In many scenarios that require the use of a UGV, the UGV is unable to rely on a priori knowledge of the terrain and obstacle locations in order to navigate and requires a perception-based sensor to locate hazards and plan a path around them. Nonlinear model predictive control (NMPC) is able to perform path planning while implementing constraints in order to maintain safe operating conditions for the vehicle.

The task of getting the UGV from point A to point B safely while avoiding obstacles has been addressed in different ways. Park, et al used a parallax-based obstacle avoidance scheme; the vehicle assumed a constant velocity and stationary obstacles. In Liu, et al, a safe region was determined from lidar data, which allowed the position of the UGV to be constrained. Again, constant speed and stationary obstacles were assumed. Abbas, et al compared two obstacle avoidance techniques, both requiring a reference

trajectory. Jiang, et al used a linear time-varying MPC and incorporated a single dimension artificial potential field method; it considers a single static or dynamic obstacle. Eick's work is the most similar to this work. A pattern search algorithm was used for optimization and a hard constraint was placed on obstacle avoidance. A single static obstacle was considered in this work.

The approach presented in this work uses a particle swarm pattern search optimization routine. Also, since in many cases the obstacles are not static, the problem of dynamic obstacles is addressed by using point cloud data to track obstacles and by incorporating constant velocity models for the obstacles into the prediction equations of the NMPC.

First, the vehicle model used to design the NMPC is discussed. Then, the optimization technique is addressed. Next, the obstacle tracking algorithm is explained. The simulation environment, setup, and results are discussed and are followed by the conclusion.

Vehicle Model

The vehicle model used in the formulation of the NMPC incorporates the lateral dynamic bicycle model [6], shown in figure 1 and characterized by equations (1-5) and equations (7-8), and the Pacejka Magic tire model [7], described in equation (6). Unlike the kinematic bicycle model, the lateral dynamic bicycle model does not make the zero slip angle assumption at the wheels. Another assumption of the bicycle model is zero vehicle roll, which is not the case; however, rollover is assumed to not be an issue in this work. The Pacejka Magic tire model is classified as a semi-empirical tire model. Joining these two models provides a sufficient amount of accuracy for the NMPC to predict the vehicle's motion.

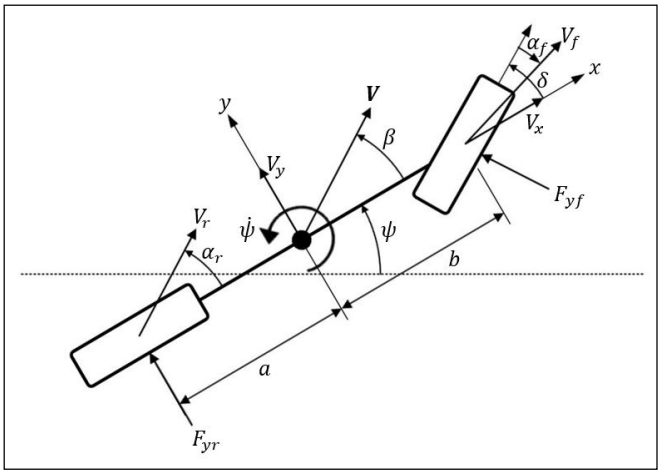

Figure 1: Bicycle model.

The vehicle state and dynamic equations are

$$\boldsymbol{x} = \begin{bmatrix} X & Y & \psi & \dot{\psi} & V_y \end{bmatrix}^T \tag{1}$$

$$\dot{X} = V_x \cos\psi - V_y \sin\psi \tag{2}$$

$$\dot{Y} = V_x \sin\psi + V_y \cos\psi \tag{3}$$

$$\ddot{\psi} = \frac{1}{I_{zz}}\left(aF_{yf}\cos\delta - bF_{yr}\right) \tag{4}$$

$$\dot{V}_y = \frac{1}{m}\left(F_{yf}\cos\delta + F_{yr}\right) - \dot{\psi}V_x \tag{5}$$

where X, Y are the global position, $\psi, \dot{\psi}$ are the yaw and yaw rate, respectively, and V_y is the body-fixed lateral speed. As previously mentioned, the lateral forces are calculated using the Magic tire model, which is

$$F_{yi} = -D\sin[C\tan^{-1}\{B\alpha_i - E(B\alpha_i - \tan^{-1}(B\alpha_i))\}] \tag{6}$$

where i is f or r, respectively corresponding to the front or rear axles, and α_f, α_r are the slip angles at the wheels defined by

$$\alpha_f = \tan^{-1}\left(\frac{V_y + a\dot{\psi}}{V_x}\right) - \delta \tag{7}$$

$$\alpha_r = \tan^{-1}\left(\frac{V_y - b\dot{\psi}}{V_x}\right) \quad (8)$$

The parameters B, C, D, E in equation (6) correspond to the stiffness factor, this determines the slope at the origin; the shape factor, this limits the range of the sine function; the peak lateral force; and the curvature factor, this sets the curvature at the peak value and adjusts the horizontal position of the peak value, x_m. These are illustrated in figure 2.

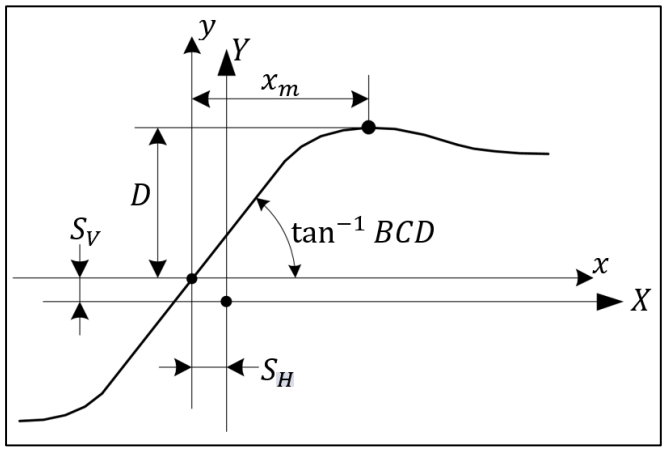

Figure 2: General tire curve.

The Y-axis is the lateral force, F_y, at the tire, and the X-axis is the slip angle, α, at the tire. The offsets S_V and S_H are due to conicity and ply-steer effects [7].

The vehicle states are measured from the outputs of a 6 degree of freedom, loosely-coupled, GPS/INS extended Kalman filter (EKF). The EKF outputs the measurements in the ENU navigation frame. One combination of real sensors used to test the EKF is a u-blox EVK-M8T with a single antenna and a Crossbow IMU440. An Animatics Smart Motor SM3430D-PLS2 attached to the steering column of the vehicle is used for the measurement of the steering wheel angle. At the beginning of each run, the steering angle is set to zero. The steer angle at the tires is found by dividing the steering wheel angle by the steer ratio.

NMPC Algorithm

The control inputs of the NMPC algorithm are the desired steer angle, δ, and the desired longitudinal velocity, V_x. The steer angle is projected over the prediction horizon using a parabolic model.

$$\delta(t) = u_0 + u_1\left(\frac{t}{t_h}\right) + u_2\left(\frac{t}{t_h}\right)^2 \quad (9)$$

where u_0, u_1, u_2 are the control parameters to be optimized, t_h is the prediction horizon in seconds, and $t = 0, \Delta t, 2\Delta t, \ldots, t_h$. From this model, the predicted vehicle model trajectory is also parabolic. The model may be of a higher order if a different form of predicted path is desired; however, this type of path is sufficient for a control rate of 10 Hz. Using a model of this nature also reduces the number of optimization parameters, whereas a typical prediction strategy is to compute a steer angle or desired curvature at each time step. Therefore, the control input vector is

$$\boldsymbol{u} = [u_0 \quad u_1 \quad u_2 \quad V_{x,des}]^T \quad (10)$$

The control input vector is optimized using a particle swarm pattern search approach. Since the dynamic equations are nonlinear, there is no closed form solution; therefore, the equations are numerically integrated using Runge-Kutta 4th order integration.

Particle Swarm Pattern Search

The particle swarm pattern search algorithm [8] minimizes the objective function

$$J = \frac{1}{2}\int\left(\kappa d_t^2 + \zeta\left(\frac{1}{\min d_{o,i}}\right)^2 + \rho e_{norm}^2\right) \quad (11)$$

subject to:

$$|\delta(t)| \leq \delta_{max} \quad (12)$$

$$|\Delta\delta(t)| \leq \Delta\delta_{max} \quad (13)$$

$$0 \leq V_{x,des} \leq V_{x,max} \quad (14)$$

where d_t is the distance to the target, κ is the weight on d_t, $d_{o,i}$ is the distance from the vehicle to each obstacle i, ζ is the weight on the inverse of the minimum distance to an obstacle, e_{norm} is the normal error of the vehicle calculated from a straight line between the starting point and the target point, ρ is the weight on the normal error, δ_{max} is the maximum allowable steer angle, $\Delta\delta_{max}$ is the maximum allowable steer rate, and $V_{x,max}$ is the maximum longitudinal speed.

The pattern search portion of the algorithm performs an element by element adjustment on the control vector. The pattern search matrix is the matrix along which the search is performed. It is of size $m\times(2m+1)$, where m is the number of control inputs. The search is performed along the columns of the matrix. For the control vector in equation (10), the pattern search matrix is

$$P = [\boldsymbol{I}(4\times4) \quad -\boldsymbol{I}(4\times4) \quad \mathbf{0}(4\times1)] \quad (16)$$

where $\boldsymbol{I}$ is the identity matrix and $\mathbf{0}$ is the zero vector. Based on equation (16), the pattern search increases u_0, then u_1, and so on. The step size each parameter is adjusted by is initialized by the user. After adjusting the input vector, the cost is calculated along the trajectory; constraints are also evaluated along the trajectory. If the calculated input generates a trajectory that satisfies the constraints and successfully lowers the cost, the input is updated as the calculated input; otherwise, the step size is decreased. The process is continued until the step size is below a tolerance value set by the user. If, during the process, a set of control inputs results in a constraint violation, the cost for that set of control inputs is set to infinity; this is to reassure the user that the set of control inputs will not be selected as an optimal solution. The pattern search algorithm is outlined in algorithm 1, where γ is the step size and θ is the reduction factor, a value between 0 and 1. The pattern search algorithm alone returns a local minimum; however, when partnered with the particle swarm algorithm, a global minimization may be produced.

While $\gamma_k > \gamma_{tol}$
 While $k \leq 2m+1$
  Calculate cost and evaluate constraints
  If $J(u_k + \gamma_k P(:,k)) < J(u_k)$ & 0 constr. viol.
   $u_{k+1} = u_k + \gamma_k P(:,k)$;
   $k = 2m$;
   search = true; (success)
  else
   $u_{k+1} = u_k$;
  end
 end
 If search == true
   $\gamma_{k+1} = \theta\gamma_k$;
 end
end

Algorithm 1: Pattern search pseudocode.

The particle swarm algorithm allows the user to make multiple initial guesses, whereas the pattern search alone uses only one initial guess. Each particle is fed through the pattern search algorithm, then the particle with the minimum cost is accepted as the final solution. Each particle is initialized randomly from a uniform distribution. Given equation (16), the first particle was set to zeros, and the second particle had zeros for the first 3 control inputs and V_{max} for the final input; however, if these initializations violated a constraint, they would be initialized randomly like the rest of the particles. In order to promote convergence, the particles are continually initialized until the initialization does not break a constraint.

Each particle has a position (the control input values) and a velocity associated with it. At the end of a particle's optimization, the particle position and velocity are updated by

$$u^i(t+1) = u^i(t) + v^i(t+1) \quad (17)$$

$$v^i(t+1) = \iota(t)v^i(t) + \mu\omega_1(t)\left(y^i(t) - u^i(t)\right) + \xi\omega_2(t)\left(\hat{y}(t) - u^i(t)\right) \quad (18)$$

where $v^i(t+1)$ is the velocity of particle i, $\iota(t)$ is known as the inertial weighting factor, μ is the cognition parameter, ξ is the social parameter, $\omega_1(t)$ and $\omega_2(t)$ are random numbers drawn from the $(0,1)$ uniform distribution, $y^i(t)$ is the best position of particle i, and $\hat{y}(t)$ is the best position among all the particles s.

Initialize particles to satisfy constraints. Select minimum to represent initial best input.
While $\gamma_k > \gamma_{tol}$
 For $i = 1{:}\, s$ (number of particles)
  Perform Pattern Search (u_{pat})
  If $J(u_{pat}) < J(y_k^s)$ & 0 constr. viol.
   $y_k^s = u_{pat}$;
   If $J(u_{pat}) < J(\hat{y})$
    $\hat{y} = u_{pat}$;
   end
  end
  Update particles with equations (17-18)
 end
 If search == true
    $\gamma_{k+1} = \theta\gamma_k$;
 end
end

Algorithm 2: Particle swarm pattern search pseudocode.

The particle swarm pattern search combined algorithm is described in algorithm 2. Computation time may become burdensome for a large number of particles; for the NMPC, 8 particles produced a sufficiently global solution while still maintaining a low computation time.

Multi-Target Detection and Tracking

Detection and tracking of multiple moving targets has been a widely studied topic for many years. Modern applications include video surveillance [9], robot navigation, and aircraft traffic control. Since Bar-Shalom's introduction of the probabilistic data association filter [10] in the 1990s numerous solutions of the tracking problem have been proposed for use with a continuously expanding field of perception equipment, including radar, laser scanners and cameras. Every sensing system brings its own set of strengths as well as caveats to the tracking problem.

For this work, the Velodyne VLP-16 laser scanner atop the vehicle in figure 13 is used to provide spatial data about the environment, including both static and dynamic objects, surrounding a UGV. The sensor provides sixteen vertical channels of 360-degree range measurements with sub-centimeter accuracy at scan speeds up to twenty hertz, making the device well-suited for capturing moving objects from the scale of a human to the scale of a large building. Utilizing this heap of data to quickly and efficiently decipher the spatial dynamics of the surroundings can be partitioned into three tasks: segmentation, data association, and target estimation. These three tasks are the focus of the discussion in the following sections.

Segmentation

To extract meaningful information from the range and angle measurements returned by a single laser scan, the data must be segmented into spatially significant subsets. Often these subsets take the form of primitive surfaces such as planes or cylinders, but a more common initial approach to dividing the data is clustering regions of points that appear geometrically consistent by some user-defined criteria. For example, two vehicles inside an empty parking lot could be described as two locally dense masses of points from observation of the scanner output. Of course, scanners with different configurations will provide data sets of differing qualities, but the methodology to

segmenting laser scanner data is easily generalized to most range-bearing sensing devices.

In this work two different segmentation approaches are taken to separate the points belonging to different physical bodies captured by the VLP-16. The first approach is based on a 2.5D grid projection of the data points that is easy to work with. Each measurement produced by the scanner consists of a range, a horizontal angle and a vertical angle that locate a single point in 3D space. When a complete scan arrives from the sensor, the points are transformed from polar to Cartesian coordinates and binned into a grid of cells with user-defined grid dimensions and cell widths. Next, for each grid cell containing points the height difference between the minimum and maximum points within the cell is calculated and compared with a threshold value. Cells which exceed this threshold are marked as occupied, while the remaining cells are marked as free. For vehicle navigation it is useful to set the height difference threshold to a physical constrain of the vehicle such as the maximum height of a traversable obstacle.

Two important navigation products of the grid arrangement are discretized knowledge of occupied space around the UGV and an image, formed by the occupied cells, of a top-down view of the vehicle's surroundings as shown in figure 3a. If the motion of the UGV and any dynamic objects around it is assumed to be planar, then the 2D image can be segmented into local clusters representing potential moving and trackable targets. Tools from image processing can be applied here, including k-means clustering and tree search structures. For simplicity and robustness, every group of connected cells in the grid is assumed to represent a single physical entity, and an N8 neighborhood search is propagated through the grid to find these groups. An N8 neighborhood consists of the eight cells surrounding a cell that is not located on the grid edge. To take into account the possible connections of branching elements that were missed in the vertical gaps of the scanner data, small clusters within a threshold proximity to larger clusters are integrated into the larger clusters. In figure 3b clusters are identified by their geometric centers, the detection information later sent to the tracking algorithm.

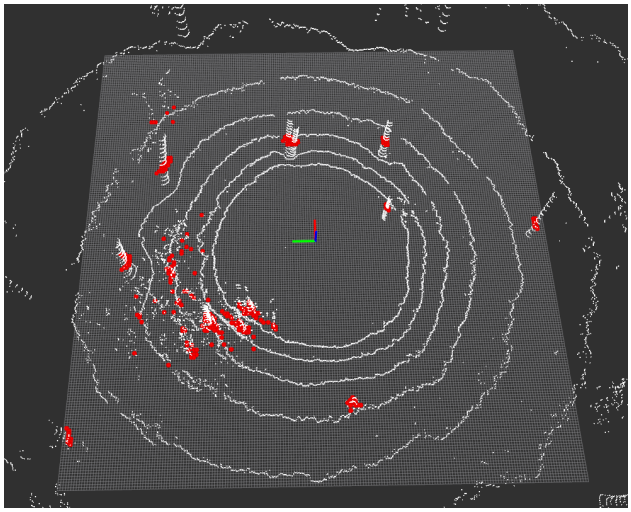

Figure 3a: Height map with occupied cells (red) corresponding to tree locations.

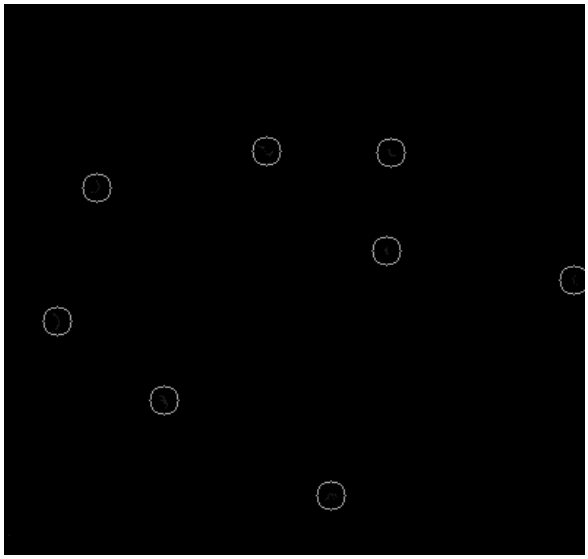

Figure 3b: Cluster centers found with height map method.

The second segmentation approach is adopted from Stachniss and Bogoslavskyi's method for segmenting sparse point clouds [11]. An advantage

of their algorithm over the previous is that it works directly with a depth image formed by inserting the data points into an array with number of rows matching the vertical channels of the scanner and number of columns matching the desired horizontal resolution of the scan. Avoiding transformations from polar to Cartesian coordinates can produce significant performance gains considering the typically large number of points found in a single scan. To prevent interference from ground points with the remainder of the segmentation process, all likely ground points are separated from the data. Figure 4 shows the angular threshold α used to check the gradient disparity between vertical channels of the Velodyne sensor. Once the ground is removed, a second angular threshold β is applied in both the row and column directions of the depth image to find gaps between point clusters. The method assumes that most physical surfaces in the scene are representable by relatively large β values, and small values indicate large spatial gaps between points as shown in figure 5. Horizontal and vertical searches for cluster groups are performed with N4 neighborhoods.

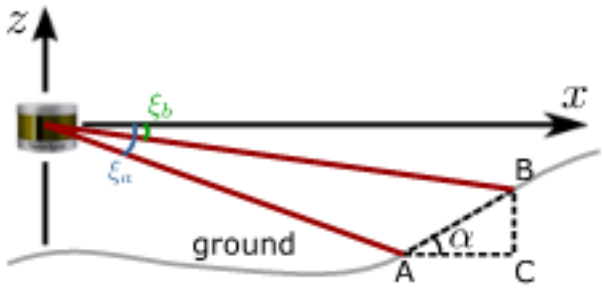

Figure 4: Ground points removal threshold angle

One disadvantage of this segmentation method is that planar surfaces at sharp angles to the scanner may be detected as multiple separate objects. Unlike the depth image ground removal strategy, the ground in the height map is removed implicitly during the cell classification stage; however, while the height map segmentation approach is dependent on a discretized 2D image of the original scan space, the depth map operates on the complete scanner data. Performance of the height map is heavily dependent on user settings of the grid dimensions and cell width. Another inherent difference between the two algorithms is the effect of extremely sparse points, such as those encountered in foliage and ground vegetation, on target detection. The height map can be made resistant to such noise by tuning the grid cell widths to blend the sparse data into connected chains, but the depth map is not so easily adjusted due to the radial divergence of the laser beams as distance from the sensor increases. In other words, introducing bins in the depth image by inflating the image columns could significantly reduce segmentation detail at long distances.

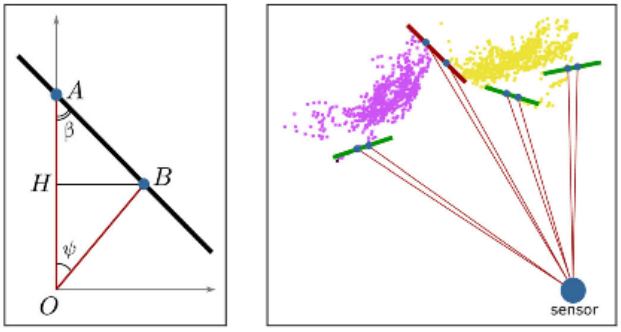

Figure 5: Point clustering threshold angle [11]. The points connected by a red line are in two different clusters.

The authors of the depth clustering algorithm kindly provide open source C++ code with their implementation written for the Robot Operating System (ROS) [11]. After several test data sets and a few modifications to the software package, the segmentation preforms well and requires minimal computing resources to function. Figure 6 shows the software's ability to detect trees inside a small park. Because depth image clustering allows for the dedication of more resources to the next component of the target tracking system, it was the detection algorithm of choice for the UGV in this work.

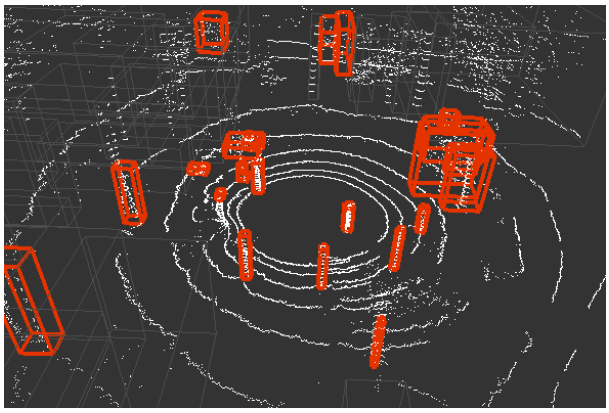

Figure 6: Objects (trees) detected by depth image clustering method.

Trimming of Detected Objects

A few simple filters are applied to the output of the segmentation algorithm to limit the scope of the target tracking as well as attenuate much of the detection noise from experimentally determined noisy detection regions. First, an effective radius around the UGV truncates object detection measurements, which leave the detection stage as 3D object centers. Next, a height threshold is applied to the center measurements given assumptions of the operating environment. For instance, tree canopies tend to produce erratic detections due to the absence of a closed surface, so cropping detections approximately above a wooded area's canopy line can drastically reduce false detection clutter. Finally, if a priori knowledge of target size is available, geometric dimensions of the detected objects can be restricted to a user-defined range.

The Kalman Filter and Gating

Detection sources could be a number of different reflective surfaces, including ground personnel, moving vehicles, and even static objects such as lamp posts and fire hydrants. For a simple and general tracking framework that works with a variety of unclassified objects, the constant velocity kinematic Kalman filter (19) was chosen as the object motion estimator. Estimation can be improved with knowledge about the model being tracked if an object classification scheme is implemented alongside the position measurements of each detected object. In the case of a moving vehicle, a bicycle car model such as the one in figure 1 and equations (1-5) could be adopted instead.

$$A = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{19}$$

The main mechanism behind pairing of incoming measurements to tracking channels engaged in estimating the positions and velocities of detected objects is the validation gate. A simple form of a validation gate is the Euclidean distance between a current estimated position and current measurement. In this work a statistical approach is taken to gating two-dimensional measurements by using a confidence level α [12], the expectation that α percent of measurements will lie outside of the validation gate, and the innovation covariance in equation (20) to define an ellipse around an object's estimated position. The confidence level is used to calculate a threshold χ^2-distributed value γ that represents the scale of an ellipse and is compared to the Mahalanobis distance [12] between the measurement and estimated object position as shown in equation (21) to determine the validity of a measurement-tracker pair. If the position and velocity states of an object are assumed to be uncorrelated, then this ellipse is a circle. Valid measurement-tracker pairs lie inside the γ threshold and are found in the next stage of the tracking problem, data association.

$$S_k = H_k P_{k|k-1} H_k^T + R_k \tag{20}$$

$$(\vec{x} - \vec{\mu})^T S^{-1} (\vec{x} - \vec{\mu}) \leq \gamma(\alpha) \tag{21}$$

Data Association

Finding valid measurement-tracker pairs can be a non-trivial assignment problem, especially in cases of high measurement clutter where multiple measurements may fall within the validation gate of a single active tracking channel. Here a tracking channel refers to a single instant of the previously mentioned constant velocity Kalman filter initialized with the measurement of a detected object and estimating the position and velocity of that object. Figure 7 is a representative outline of the data association routine developed for this work.

The routine has a hierarchal structure, where active trackers and measurements are conditioned prior to association to combat ill-formulated assignment problems arising from undesirable scenarios and to increase the efficiency of the association stage. Before any further explanation of the association stage, a description of the mechanics behind the tracking channels is due. For the first set of measurements received by the association stage of the multi-target tracking system, a tracking channel is initialized for each measurement. At this point there are the same number of active tracking channels as there are measurements. A tracking channel, or tracker, may be either ACTIVE or INACTIVE. For every subsequent set of measurements, each measurement must either be matched to an active tracker or used to activate a new tracker. Trackers that are paired with an incoming measurement receive a tick in their ASSOCIATION counter. Unpaired trackers receive a DISSOCIATION tick. If a tracker is not matched with a measurement for a user-defined threshold of consecutive measurement updates, that tracker is deactivated. In addition, trackers that are matched consecutively less than a user-defined number of update cycles can be deactivated more easily than a tracker with a longer continuous history.

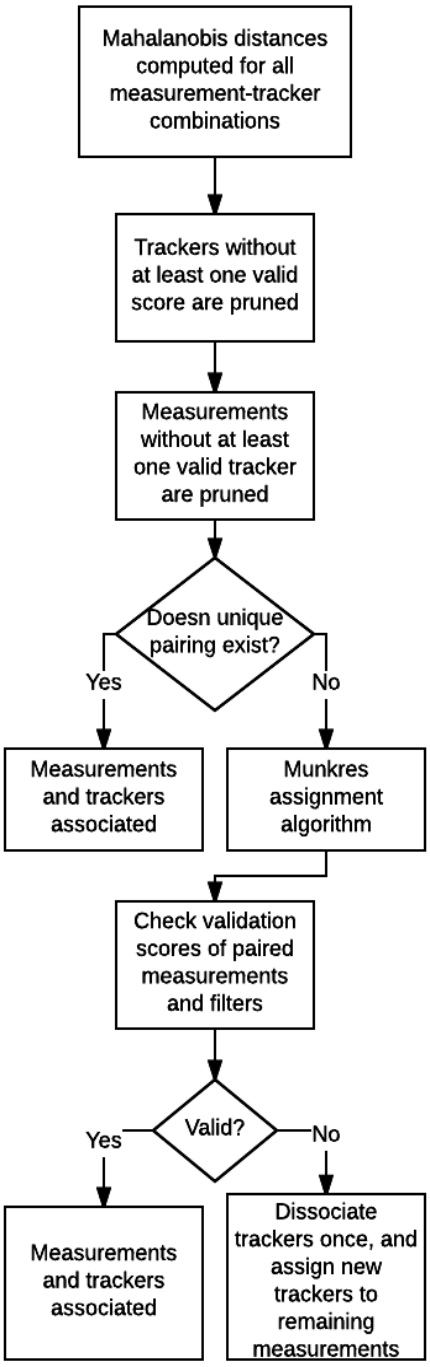

Figure 7: Data association routine.

As shown in figure 7 the first step of the hierarchal association stage is computation of the Mahalanobis distances between all measurements and all active trackers. Distance data can be

visualized as a matrix where rows represent measurements and columns represent active trackers. All trackers that do not hold at least one distance score less than or equal to γ in their respective columns are removed from the current round of matching. Similarly, all measurements without a score less than or equal to γ in their respective rows are removed from the matching process as well. Next, a naïve assignment is attempted between remaining trackers and measurements by assigning each tracker to the corresponding measurement of that tracker's minimum distance score. If this pairing is unique for all trackers, then the association stage is complete. Otherwise, the association task is handed to a Munkres assignment process [13].

The Munkres algorithm applies a series of sequential modifications to the elements of a 2D matrix with arbitrary numbers of rows and columns until the elements of the smaller dimension are matched uniquely with the same number of elements in the larger dimension, and the sum of the values at the intersections of these row-column pairs is the minimum possible sum. For the matrix of scores, each element in the smaller dimension is paired with a unique element in the larger dimension, and the sum of the scores of these measurement-tracker pairs is a global minimum. If nine measurements and eleven trackers are passed to this algorithm, nine unique measurement-tracker assignments are created. These assignments are identified by the zeros in figure 8. Two of the trackers do not have zeros in their columns and are not paired with any measurements.

Pre-conditioning of the measurements and trackers at the beginning of the association stage helps avoid scenarios where a tracker that has lost its target joins in the Munkres assignment phase trackers whose targets are still providing measurements. If this tracker in question is in the dissociation process but still active, and if it has lost its target, then its distance scores to the remaining measurements will likely be relatively high. The Munkres algorithm will choose a globally lowest score for each tracker if the number of trackers is equal to or less than the number of measurements. In the event that the dissociating tracker is paired with another currently active tracker's measurement for the sake of reducing the sum of distance scores, then the other tracker will be assigned to a different measurement. Assignments are followed by a final validation check. Pairs that pass are considered associated, and the corresponding measurement update is applied to those measurement-tracker pairs. On the other hand, pairs that do not pass do not receive a measurement and enter the dissociation phase. This example demonstrates a tracker being kicked out of its valid pair by a dissociating tracker, hence the need for preconditioning the trackers and measurements prior to assignment.

```
-1,-1, 0,-1,-1,-1,-1,-1,-1,-1,-1,
-1,-1,-1, 0,-1,-1,-1,-1,-1,-1,-1,
-1,-1,-1,-1,-1,-1, 0,-1,-1,-1,-1,
-1,-1,-1,-1,-1,-1,-1, 0,-1,-1,-1,
-1, 0,-1,-1,-1,-1,-1,-1,-1,-1,-1,
-1,-1,-1,-1, 0,-1,-1,-1,-1,-1,-1,
-1,-1,-1,-1,-1, 0,-1,-1,-1,-1,-1,
-1,-1,-1,-1,-1,-1,-1,-1, 0,-1,-1,
-1,-1,-1,-1,-1,-1,-1,-1,-1,-1, 0,
```

Figure 8: Munkres assignment algorithm. Zeros represent assignment of nine measurements to nine unique trackers.

Figure 9 shows the tracking algorithm developed for ROS during the course of this work. The scene is a small park with tall trees and uneven terrain. Although the ground is not planar here, the depth clustering algorithm effectively removes the ground points from the scanner data and detects the trees. Red spheres represent the detected point clusters, while the blue ellipses around the spheres are centered on the target position estimates and represent the γ-threshold gating regions for new measurements.

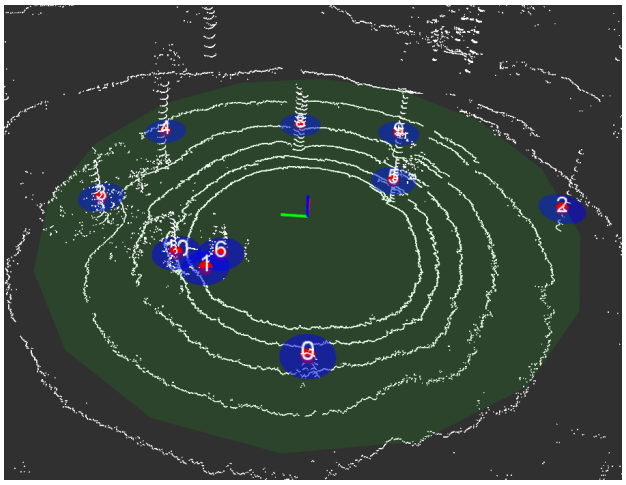

Figure 9: Multi-target tracking. Tree point clusters (red) and estimates of position (blue centers) with tracker IDs are shown.

SIMULATION

Gazebo, a simulation engine from the Open Source Robotics Foundation, is used to combine and test the developed obstacle tracking and avoidance algorithms inside a safe and controlled environment. A 2004 Infiniti G35 sedan was chosen for the vehicle model, with close approximations of the real vehicle's geometric, kinematic, inertial, and performance properties acquired from the lab's existing G35 data collection platform. The Robot Operating System (ROS) is used to interface the vehicle model inside the simulation environment with sensors, controllers, and the obstacle avoidance software. Controllers are abstracted to mimic a vehicle interface that can be replaced with a real vehicle in the future without modifying the navigation software. Sensors, including a VLP-16 lidar, GPS, and 9-DOF IMU, are virtual, but their software sockets are easily replaceable with real hardware.

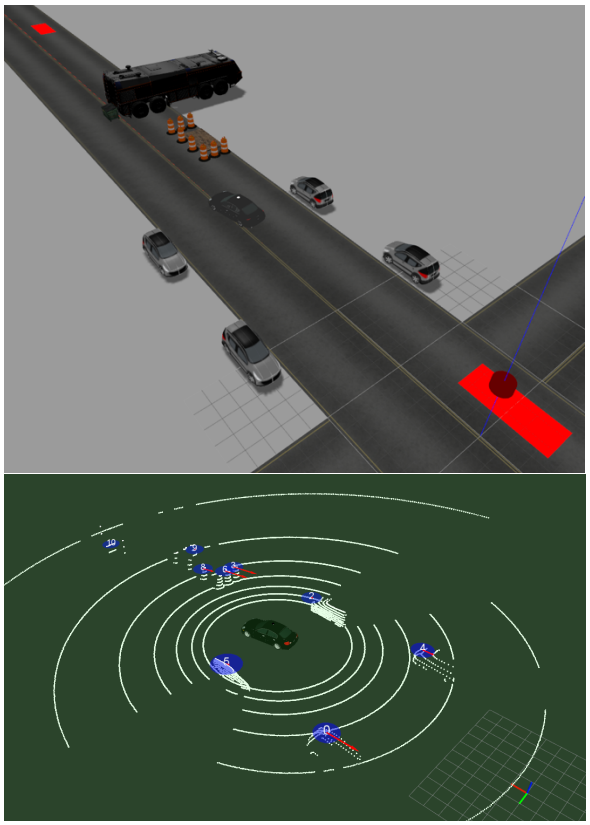

Figure 10: Gazebo environment (top) with red rectangles representing the start and goal positions of the vehicle. Multi-target tracking (bottom) in Rviz, where tree point clusters (red) and estimates of position (blue centers) with tracker IDs are shown.

Two obstacle avoidance scenarios are simulated in Gazebo. One scenario requires the vehicle to pass around a static obstacle placed between the vehicle's start and target positions. In the second scenario, the static obstacle is replaced with a dynamic one moving perpendicular to the vehicle's desired path. Because both the position and velocity of the obstacle are provided by the tracker, the NMPC can predict the future state of the moving obstacle and produce a trajectory that avoids collision. The simulation parameters are listed in table 1, where each parameter value is listed in brackets beside the parameter's name.

Mass $[m]$	1728.15 kg
Front axle to c.g. $[a]$	1.3675 m
Rear axle to c.g. $[b]$	1.4815 m
Yaw inertia $[I_{zz}]$	2400 kg m^2
Magic parameter $[B]$	9.55
Magic parameter $[C]$	1.3
Magic parameter $[D]$	6920
Magic parameter $[E]$	0
Max steer angle $[\delta_{max}]$	35°
Max steer change $[\Delta\delta_{max}]$	5°
Initial step size $[\gamma]$	1
Reduction factor $[\theta]$	0.5
Number of particles $[s]$	8
Tolerance $[\gamma_{tol}]$	0.01
Control frequency $[1/\Delta t]$	10 Hz
Prediction horizon $[t_h]$	5 s
Initial state vector	$[0 \quad 0 \quad 90° \quad 0 \quad 0]^T$
Weight on distance to target $[\kappa]$	5
Weight on inverse of min. obstacle distance $[\zeta]$	225000
Weight on normal error $[\rho]$	17
Maximum Velocity $[V_{x,max}]$	5 m/s
Final time $[t_f]$	19.87 s

Table 1: Simulation parameters.

Figure 10 shows a hybrid scenario that includes both a static and dynamic obstacle. The static obstacle is composed of eight construction barrels blocking the vehicle's lane, while the dynamic obstacle is in the form of a large truck backing into the lane from the occluded region behind the barrels. Figures 11 and 12 show that the controller, aided by the obstacle tracker, is able to guide the vehicle around both obstacles.

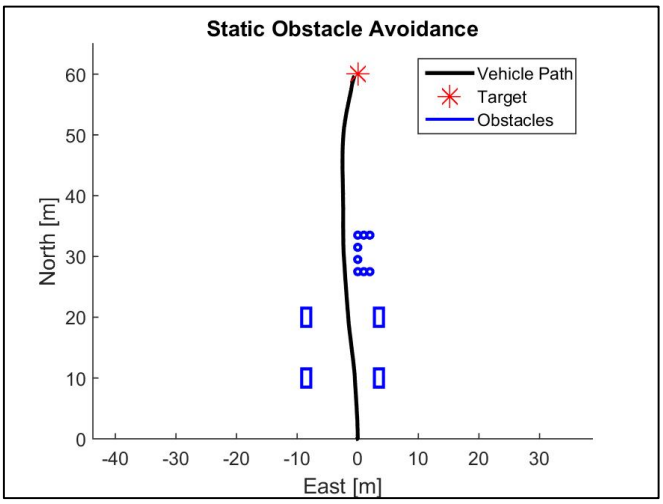

Figure 11: Static obstacle avoidance in Gazebo.

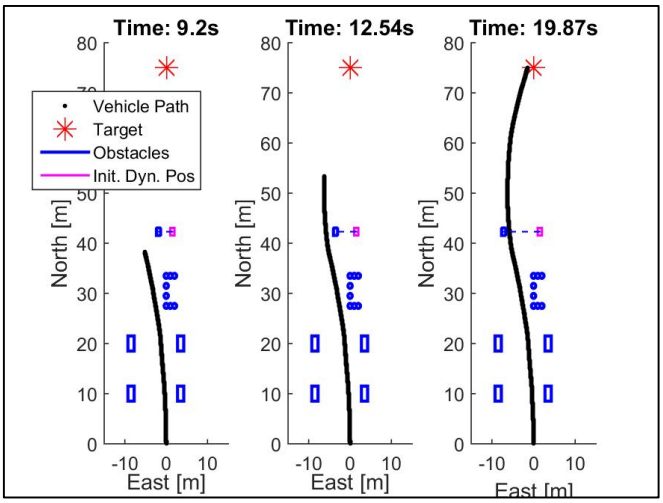

Figure 12: Dynamic obstacle avoidance in Gazebo.

CONCLUSIONS

A NMPC for avoidance of static and dynamic obstacles was presented. It incorporates a particle swarm pattern search algorithm for solving the constrained optimization problem. Point cloud data from lidar was used by a detection algorithm to find clusters of points likely belonging to unique objects, and a bank of constant velocity Kalman filters was used to track these clusters and provide

obstacle position and velocity information to the NMPC. Gazebo simulation results were presented showing successful obstacle avoidance by the virtual vehicle in real time.

FUTURE WORK

After further simulation validation of the NMPC and tracking algorithms, the software will be loaded onto the Prowler RTV off-road vehicle shown in figure 13. This vehicle is equipped with a complete drive-by-wire automation system interfaced with ROS. Actuators are operated by a microcontroller and control throttle, braking, and steering. The software of the virtual Infiniti G35 is being modified to accommodate the Prowler along with a model of its actuation system. A simulated system with a ROS interface identical to the one on the real vehicle is the goal and will allow quick and easy interchange between evaluating changes on the simulated vehicle and evaluating them on the real vehicle.

Figure 13: Prowler UGV with Velodyne VLP-16 scanner mounted to the roof.

REFERENCES

[1]Park, J., Kim, D., et al., 2009. "Obstacle avoidance of autonomous vehicles based on model predictive control". *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 223(12), pp. 1499–1516.

[2]Liu, J., Jayakumar, P., et al. "A Nonlinear Model Predictive Control Algorithm for Obstacle Avoidance in Autonomous Ground Vehicles within Unknown Environments", Defense Technical Information Center, 2015.

[3]Abbas, M. A., Milman, R., Eklund, J. M., "Obstacle Avoidance in Real Time With Nonlinear Model Predictive Control of Autonomous Vehicles", *Canadian Journal of Electrical and Computer Engineering*, vol 40, no 1, pages 12-22, 2017.

[4]Jiang, H., Wang, Z., et al. "Obstacle Avoidance of Autonomous Vehicles with CQP-based Model Predictive Control", In International Conference on Systems, Man, and Cybernetics, 2016, IEEE, pp. 1668-1673.

[5]Eick, A., Bevly, D., "Nonlinear Model Predictive Controller for an Unmanned Ground Vehicle on Variable Terrain", In ASME 2015 Dynamic Systems and Control, American Society of Mechanical Engineers.

[6]Jazar, R. N., 2008. *Vehicle Dynamics Theory and Application*. Springer.

[7]Pacejka, H. B., 2006. *Tyre and Vehicle Dynamics*. Elsevier.

[8]Vaz, A., Vicente, L., "A particle swarm pattern search method for bound constrained global optimization", Science and Business Media B.V., and Cybernetics, 2007, Springer, pp. 197-219.

[9]Rezatofighi, Seyed Hamid et al. "Joint Probabilistic Data Association Revisited." *2015 IEEE International Conference on Computer Vision (ICCV)* (2015): 3047-3055.

[10]Y. Bar-Shalom, F. Daum and J. Huang, "The probabilistic data association filter," in *IEEE Control Systems*, vol. 29, no. 6, pp. 82-100, Dec. 2009.

[11]Bogoslavskyi, Igor, and Cyrill Stachniss. "Efficient Online Segmentation for Sparse 3D Laser Scans." *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science* 85.1 (2017): 41-52.

[12]Warren, Rik, Robert F. Smith, and Anne K. Cybenko. *Use of Mahalanobis distance for detecting outliers and outlier clusters in markedly non-normal data: a vehicular traffic example*. SRA INTERNATIONAL INC DAYTON OH, 2011.

[13] Gottschalk, T. D. "Concurrent implementation of Munkres algorithm." *Distributed Memory Computing Conference, 1990., Proceedings of the Fifth*. IEEE, 1990.

Disclaimer: Reference herein to any specific commercial company, product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the Department of the Army (DoA). The opinions of the authors expressed herein do not necessarily state or reflect those of the United States Government or the DoA, and shall not be used for advertising or product endorsement purposes.