Optimization-based path planning and collision avoidance for autonomous racing

Oran Keanly[1]*, *Jacobus Adriaan Albertus* Engelbrecht[1]

[1]Department of Electrical and Electronic Engineering, Stellenbosch University, South Africa

Abstract. This paper presents a hierarchical motion planner for autonomous racing. The long-term motion planner functions offline and formulates the optimal motion plan for the entire race track. The short-term collision avoidance planner functions online and formulates a motion plan for a limited horizon ahead of the autonomous car when an obstacle is detected in the path of the vehicle. The motion planners formulate the planning problems as optimal control problems and solve the resulting optimizations using an interior point optimizer (IPOPT). Simulation experiments show that an autonomous vehicle using the motion planner is able to race around the track with minimum lap time while avoiding unexpected obstacles.

1 Introduction

Racing is an inherently competitive and high-performance activity and when extended to autonomous vehicles it creates a good environment for development in many areas including robotics, control algorithms, and computer vision. Thus, autonomous racing has attracted significant attention in both academia and industry [1]. The F1tenth project is a well-known and tested platform for autonomous racing and thus we have designed our system for use on an F1Tenth car as shown in **Fig. 1**. Included in the F1Tenth design is a LiDAR scanner, a virtual electronic speed controller (VESC), and a camera [2]. For a car to race safely, an autonomous vehicle must plan and execute a racing path, as well as avoid unexpected obstacles in the environment. Therefore, we propose a hierarchical optimization-based motion planner that will plan a global path offline as well as an online motion planner that will plan a limited horizon local path in the environment to avoid any obstacles.

Fig. 1. An F1Tenth Car [2]

* Corresponding author: 21738394@sun.ac.za

2 Background

2.1 Related word

Much of the current research in autonomous racing models vehicles using a bicycle model [3, 4]. Although there is a more accurate double-track vehicle model, such as the one used by Fors et al. [5] and Novi et al. [6], it is significantly more complex. When modelling a vehicle using the bicycle model one often considers either the kinematic model or the dynamic model. The kinematic model considers only the geometric relationship between each component and does not consider the forces or torques involved in the system. The dynamic model takes the forces and torques of the system into account when modelling the vehicle, leading to a more complex model, but one that can capture phenomena such as suspension dynamics or tyre slip. Kong et al. [7] performed a side-by-side comparison of a dynamic bicycle model and a kinematic bicycle model. In this comparison, vehicle states were recorded from a physical vehicle driving around a track. Then a series of simulations were performed where a model was initialized to a recorded state and a recorded input sequence was input to the model. Then the distance between the vehicle's actual location and the model predicted location was compared. This showed that the mean open-loop prediction error of the kinematic model is initially very similar to the dynamic model. However, after three 200ms timesteps the mean open-loop prediction error of the kinematic model was greater than the dynamic model by 30%, and after four timesteps it was 48% higher than the dynamic model.

When studying motion planning, there are many methods of autonomous motion planning such as classical planning techniques [3, 8] and learning-based [9]. Classical planning techniques such as optimization-based path planning are often used due to their accuracy, however they can be computationally expensive. The problem of trajectory planning and control for autonomous race cars has been investigated by a number of previous researchers. Heilmeier et al. [10] generated a path based on minimizing the curvature of the path. Vázquez et al. [11] used an optimization-based path planning algorithm in order to generate a minimum lap-time trajectory. Model Predictive Control (MPC) was used by both Ahlberg [8] and Sjolin and Sundberg [12] to generate a local motion plan for a short time horizon. MPC has the advantage of being able to respond to unexpected obstacles, as discussed in [12], which many optimisation-based path planning methods struggle to handle in real-time. This is due to the complexity of formulating and solving an optimization problem online and thus the need for significant computing power. This paper presents a solution to the online optimization

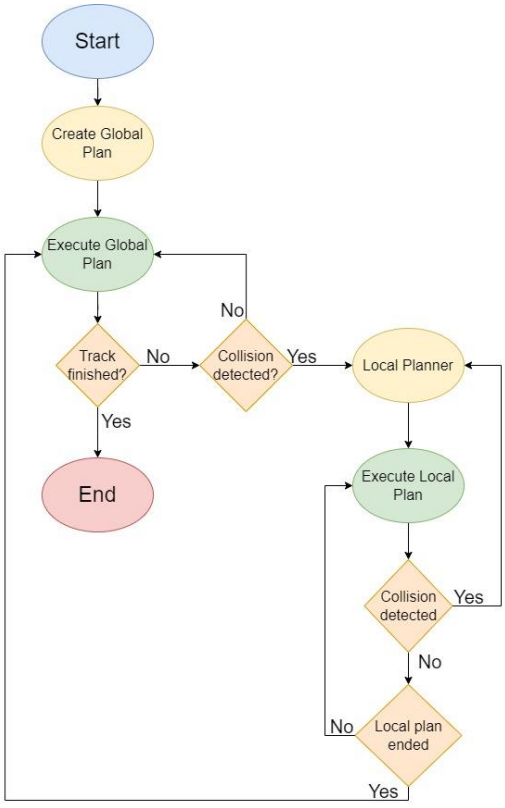

Fig. 2. System flow diagram for the proposed autonomous racing system

problem, by using simplified vehicle dynamics, as well as introducing additional constraints to the formulation of the local path planning problem.

2.2 Research Contributions

This paper presents an autonomous racing system that is computationally inexpensive and capable of racing competitively while avoiding unexpected obstacles. The autonomous racing system is comprised of the following major components:

- A long-term motion planner that can calculate a minimal lap-time motion plan for the vehicle.
- A control system to execute the motion plan.
- A collision prediction system that is capable of detecting obstacles on the racetrack and determining whether evasive action is required.
- A short-term motion planner that can calculate an optimal short-term motion plan for the car to avoid detected obstacles.

The following novel contributions are made:

- A novel collision prediction method was developed that uses 2D LiDAR measurements to detect and map unexpected obstacles on the track and propagates the position and heading of the car up to a prediction horizon time to predict collisions.
- A novel short-term motion planner was developed that uses a simpler kinematic bicycle model instead of the full dynamic model and provides a computationally less expensive method to calculate the short-term collision avoidance plan for the car.

3 Methodology

This paper proposes the methodology shown in **Fig. 2**. In the offline path planning phase, a global path plan is calculated with an optimisation problem formulated using a curvilinear bicycle model to model the vehicle. The global plan is then executed, and during this time the 2D LiDAR scanner is used to detect unexpected obstacles on the track by clustering any points in the resultant point cloud that are on the track. If an object is detected, then a check is done to see whether that obstacle will cause a collision with the car if no additional action is taken. If a collision is predicted, an optimisation problem is formulated and solved online with additional constraints on the allowable locations in order to generate a short-term local plan that can be used as a reference to avoid the obstacle and continue racing.

4 Modelling

In this section, we present the vehicle and track model used in the hierarchical motion planner. This allows us to formulate the necessary structure for the path planning problem, while showing the necessary adjustments that are made for the online short-term planner. First, we discuss the curvilinear track model, as well as the vehicle bicycle model that is formulated within the curvilinear coordinate system. Then we proceed to briefly discuss how the time-dependant model is related to a space-dependent model for the path planning process.

4.1 Curvilinear Bicycle Model

We model the car using a bicycle model with dynamic force laws [11]. This model is significantly less complex than a full double-track vehicle model. However, the inclusion of dynamic force laws still allows for the modelling of important racing phenomena such as tyre slip.

This bicycle model is not set in a global co-ordinate frame, but in a curvilinear co-ordinate system. This co-ordinate system is formulated locally with respect to the racing track, which is modelled as a curvilinear centre line and an orthogonal track width. Thus, the curvilinear co-ordinates of the car are referenced to the centre line of the track as shown in **Fig. 3**. This means that no global position or heading is considered and that the state information of the vehicle is relative to the reference track. If a global position or heading is required, it can be determined from the reference track. Similarly, curvilinear co-ordinates can be determined from global co-ordinates by projecting the global co-ordinates onto the reference track.

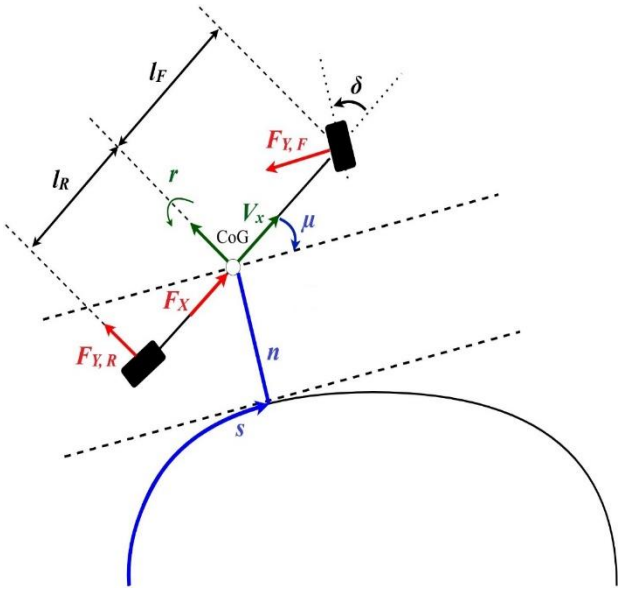

Fig. 3. Illustration of the dynamic bicycle model in the curvilinear co-ordinate system

The state information of the vehicle in relation to the track consists of the progress along the track centre line s, the orthogonal distance from the centre line n, and the local heading μ. These states develop naturally from the curvilinear co-ordinate system and the dynamics of these states are all dependent on the curvature of the centre line. However, to continue the development of the vehicle model we make the following assumptions:

- The car travels on level ground such that all vertical motion is negligible.
- Load changes can be ignored.
- Combined slip can be neglected.
- All drive train forces act at the centre of gravity.

- The vehicle is a rigid body with mass m, and a rotational inertia I_z.
- The throttle command of the car can change instantaneously.
- The steering angle of the car change instantaneously.

Using these assumptions, we can continue to develop the vehicle model and its dynamics, starting with the longitudinal velocity v_x and the lateral velocity v_y as well as the yaw rate, r. Finally, we consider the input to the vehicle model, namely the steering angle δ and a throttle command T. Thus, the state vector is defined as $\boldsymbol{x} = [s;\ n;\ \mu;\ v_x;\ v_y;\ r]$ and the input vector is defined as $\mathbf{u} = [\delta;\ T]$. **Fig. 3** shows the bicycle model and the curvilinear co-ordinate system.

Using the assumptions given above, the dynamics of the states can be derived as seen in (1) [11, 13].

$$\begin{aligned}
\dot{s} &= \frac{v_x \cos(\mu) - v_y \sin(\mu)}{1 - n\kappa(s)} \\
\dot{n} &= v_x \sin(\mu) + v_y \cos(\mu) \\
\dot{\mu} &= r - \dot{s}\kappa(s) \\
\dot{v_x} &= \frac{1}{m}\left(F_x - F_{y,F}\sin(\delta) + m v_y r\right) \\
\dot{v_y} &= \frac{1}{m}\left(F_{y,R} + F_{y,F}\cos(\delta) - m v_x r\right) \\
\dot{r} &= \frac{1}{I_z}\left(F_{y,F} l_F \cos(\delta) - F_{y,F} l_R\right)
\end{aligned} \tag{1}$$

In the equations in (1), $\kappa(s)$ is the curvature of centre line at the point along the centre line s, F_x is the longitudinal force acting on the car, and $F_{y,F}/F_{y,R}$ is the lateral force acting on the car at the front and rear wheels, respectively. The dynamics described in (1) is denoted by $\dot{\boldsymbol{x}} = f_t(\boldsymbol{x}, \mathbf{u})$, where the subscript t indicates that the dynamics are formulated as a time-domain model.

The tyre forces in equation (2) describe the interactions between the car and the ground and are an integral part of the vehicle model. We use a simplified Pacejka tyre model to calculate the lateral forces while the longitudinal force is a linear function of the throttle command T, a rolling resistance, and drag-force resistance. [14, 15]

$$\begin{aligned}
F_{y,F} &= F_{N,F} D_F \sin(C_F \tan^{-1}(B_F \alpha_F)) \\
F_{y,R} &= F_{N,R} D_R \sin(C_R \tan^{-1}(B_R \alpha_R)) \\
F_X &= C_m T - C_{r0} - C_{r1} {v_x}^2
\end{aligned} \tag{2}$$

The equations in (2) give the expressions for the forces acting on the car, where $F_{N,F}/F_{N,R}$ is the normal force on the front tyre and rear tyre, respectively, and α_F/α_R is the slip angle at the front and rear tyre. Additionally, D_F/D_R, C_F/C_R, B_F/B_R are the parameters of the simplified Pacejka model for each tyre. C_m is a scaling factor for the motor force, while C_{r0} and C_{r1} are the coefficients for the rolling resistance and the drag-force resistance, respectively.

4.2 Space-dependent Model

The vehicle model and dynamics that are described above are formulated as a function of time. However, using the same techniques that Vázquez et al. [11] presented, the equations in (1) are transformed to be a function of space, instead of time by using the transformation in (3).

$$\dot{\boldsymbol{x}} = \frac{\partial \boldsymbol{x}}{\partial t} = \frac{\partial \boldsymbol{x}}{\partial s}\frac{\partial s}{\partial t}$$
$$\Rightarrow \frac{\partial \boldsymbol{x}}{\partial s} = \frac{1}{\dot{s}} f_t(\boldsymbol{x}, \mathbf{u}) = f_s(\boldsymbol{x}, \mathbf{u}) \tag{3}$$

The equation in (3) shows the dynamics of the space-dependent bicycle model, where the subscript s shows that the dynamics are based in the space-domain. One advantage of this reformulation from the time-dependent model to a space-dependent model is that the curvature of the racetrack does not change based on the current state. If the model is space-dependent, then the curvature $\kappa(s)$ can be calculated independently of the vehicle dynamics. Additionally, because the state s becomes redundant, it can be dropped from the vehicle state space thereby reducing the complexity of the system. Thus, the new reduced state vector is given by $\bar{\mathbf{x}} = [n;\ \mu;\ v_x;\ v_y;\ r]$. One final advantage of this reformulation is that it naturally lends itself to formulating a minimum-time cost function for the lap time optimization. The time that it takes to travel from a point along the centre line s_0 to another point s_1 is given in (4)

$$\text{Time from } s_0 \text{ to } s_1 = \int_{s_0}^{s_1} \frac{1}{\dot{s}}\, d\tau \tag{4}$$

There are some disadvantages associated with this transformation, however. The additional multiplication with $\frac{1}{\dot{s}}$ in every state equation increases computational cost. Additionally, if the vehicle were to stop ($\dot{s} = 0$) this would introduce a singularity in the system. Finally, an aspect that is very applicable in our case, this space-domain model implies that when it is used for control of the system, inputs to the system should be given at a fixed sampling distance as opposed to a fixed sampling time.

5 Motion Planning and execution

In this section, the vehicle and track models are used to formulate the motion planning as an optimal control problem, which is solved using an interior-point optimizer. Additionally, we present how the resulting minimum lap-time path is executed using a Stanley controller.

5.1 Optimization-based path planning

Using the space-dependent vehicle dynamics presented in (3) and (4) we can develop an optimal control problem that represents the minimum-lap time problem. The first step necessary for this is to formulate the necessary constraints in addition to the vehicle model. The optimal control problem is formulated using the techniques presented by Vázquez et al. [11]. First, the vehicle must stay within the allowable racing track. This constraint is formulated considering the local heading of the car as well as the orthogonal distance away from the centre line (the state n) while considering the dimensions of the car as shown by (5).

$$\begin{aligned} n + \frac{L_c}{2}\sin(|\mu|) + \frac{W_c}{2}\cos(\mu) &\leq N_L(s) \\ -n + \frac{L_c}{2}\sin(|\mu|) + \frac{W_c}{2}\cos(\mu) &\leq N_R(s) \end{aligned} \tag{5}$$

where L_c is the length of the car, W_c is the width of the car, and $N_L(s)$ and $N_R(s)$ are the orthogonal width of the track to the left and right of the centre line, respectively. We denote this constraint is as $\bar{\mathbf{x}} \in X_{track}$.

Additionally, given that the vehicle model that is used does not consider the combined slip, and therefore combined tyre forces, a friction constraint is necessary in order to limit the combined maximum tyre force at any given instant. This friction ellipse constraint is given by (6).

$$\begin{aligned} (\rho_{long}F_{M,F})^2 + {F_{y,F}}^2 &\leq (\lambda D_F)^2 \\ \left(\rho_{long}F_{M,R}\right)^2 + {F_{y,R}}^2 &\leq (\lambda D_R)^2 \end{aligned} \tag{6}$$

This friction constraint uses the motor force F_M which is assumed to be applied equally between the front and rear tyres such that $F_{M,F} = F_{M,F} = \frac{F_M}{2}$. This assumption is based on the assumptions that were made when developing the vehicle model in section 4. Although it becomes less accurate when the car is undergoing extreme acceleration and deceleration, it is still applicable for the purpose of constraining the maximum tyre forces. The shape of the friction ellipse is determined by ρ_{long} while the maximum combined tyre force is determined by λ. This constraint ensures that the minimum lap time optimization results do not cause the vehicle tyres to experience slip or saturation. We denote this friction ellipse constraint as $\bar{\boldsymbol{x}} \in X_{friction}$.

Finally, the steering angle and throttle command inputs are constrained based on the physical limitations of the vehicle. This constraint is denoted by $\boldsymbol{u} \in A$.

Using these constraints, an optimal control problem can be formulated that uses the space-based dynamics in (4) to compute the minimum lap time path. To formulate the optimization problem, the pre-determined track centre line is discretized, as well as the continuous space-based state equations in (3). s is discretized with a discretization distance of Δs using a forward Euler integrator with a spatial integration element. Therefore, the discrete space-based state dynamics can be expressed by (7)

$$\begin{aligned} \boldsymbol{x}_{k+1} &= f_s^d(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ &= \boldsymbol{x}_k + \Delta s\, f_s(\boldsymbol{x}_k, \boldsymbol{u}_k) \end{aligned} \tag{7}$$

The centre line is discretized with N steps and Δs is chosen such that $\Delta s(N+1)$ is equal to the total arc-length of the centre line. The cost function which seeks to minimize the lap time then becomes:

$$Time = \sum_{k=0}^{N} \frac{\Delta s}{\dot{s}_k} \tag{8}$$

However, a regularization term is included in the cost function that penalizes the difference between the kinematic side slip angle and the dynamic side slip angle. This is intended to minimize abrupt changes or inconsistencies in the vehicle's lateral motion in the generated trajectory plan. This penalization term is denoted by $B(\boldsymbol{x}_k)$.

The lap time optimization problem is given by:

$$\begin{aligned} &\min_{\boldsymbol{X},\boldsymbol{U}} \sum_{k=0}^{N} \frac{\Delta s}{\dot{s}_k} + B(\boldsymbol{x}_k) \\ s.t. \quad &\boldsymbol{x}_{k+1} = f_s^d(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ &\boldsymbol{x}_0 = \boldsymbol{X}(0) \\ &\boldsymbol{x} \in X_{track} \\ &\boldsymbol{x} \in X_{friction} \\ &\boldsymbol{u} \in A \\ &k = 0, \dots, N \end{aligned} \tag{9}$$

This problem is formulated offline using *Pyomo* [16, 17] and the resulting nonlinear optimization is solved using *Ipopt*.[18]

5.2 Path Execution

The results from the optimisation process yields the minimum lap time path that the car can take, as well as the necessary states and inputs required to execute this path. It is important to note that the path planning results are based on the track centre line s. Therefore, when implementing a control strategy for the car, it is necessary to either transform the results to the time domain or the control system must be based in the spatial domain. We choose to formulate our control system in the spatial domain, where the inputs are given at a fixed Δs as opposed to a Δt.

The velocity reference is used directly from the path planning results as. We use a Stanley control algorithm to generate the necessary steering inputs to follow the reference path. [19] The Stanley control formulation is shown in (10), where φ is the heading error of the car, K is the gain, e is the cross-track error and v is the speed of the car.

$$\delta = \varphi + \tan^{-1}\left(\frac{K \times e}{v}\right) \tag{10}$$

6 Collision Avoidance

In this section we present how unexpected obstacles within the track are identified, modelled, and if necessary, the local planner creates short-term plan to avoid the obstacle.

6.1 Collision Detection

In order to avoid unexpected obstacles on the track it is first necessary to detect them. We use the Lidar scanner on the car to create a 2D scan of the racetrack in front of the car. The resulting point cloud from this scan is projected onto the track and converted to the same curvilinear co-ordinate system as the vehicle. This allows us to naturally distinguish between the points that make up the track border and the points that are within the track that may be

an unexpected obstacle. We do this by examining the orthogonal distance each point is away from the centre line of the track and if this distance is smaller than the track width then those points may be indicative of an obstacle in the track. Following this, the points from the point cloud that are within the track are clustered using the DBSCAN (Density-based spatial clustering of applications with noise) [20]. The resulting labels from the clustering process are treated as unique obstacle in the track. Each obstacle is modelled in the curvilinear co-ordinate system as an exclusion zone at each discretized s point on the centre line. For every discretized point along the centre line that the obstacle occupies, there is an equality as shown in (11).

$$O_L(s) \leq O(s) \geq O_R(s) \tag{11}$$

If an obstacle(s) is detected by the scan, a collision predictor is activated to determine whether a short-term plan is necessary in order to avoid the obstacle(s). The collision predictor uses the current position and heading of the car, as well as the control inputs according to the reference plan that is currently being executed, to propagate the position and heading of the car up to a prediction horizon time. It is important to note that if the vehicle is currently executing a local plan, and the local plan would end within the prediction horizon, then it is assumed that the car will return to the global path after the local plan has ended using the Stanley controller. If, in the horizon time, the car would occupy the same space as an obstacle, a short-term plan is necessary to avoid the obstacle.

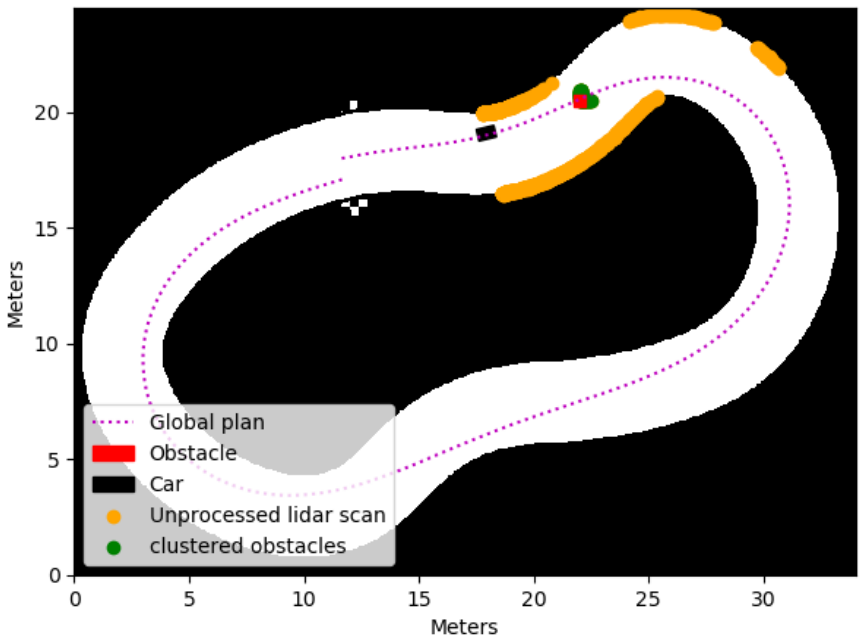

Fig. 4. An example of collision detected with an obstacle in the track.

6.2 Kinematic Bicycle Model

The dynamic bicycle model that was used for the global path planning process is a complex model, and although it is a good approximation of the true vehicle dynamics, the use of this model in a real-time path planning process would necessitate significant computation power

that may not be available. Therefore, the simpler kinematic bicycle model is used in the local planner. Although this model is not as accurate as the dynamic bicycle model, is sufficient to use for the short-term plan. Figure 5 shows the kinematic bicycle model, which is based purely on the geometry of the car and does not model the interaction between the tyres and the ground. The states of the kinematic vehicle $\boldsymbol{x}_{Kin}$ are described by the vehicle position s and n, the local heading μ. The inputs to the system $\boldsymbol{u}_{Kin}$ are the velocity v and the steering angle δ. The dynamic equations of the kinematic model are given by:

$$\begin{aligned} \dot{s} &= \frac{v_x \cos(\mu) - v_y \sin(\mu)}{1 - n\kappa(s)} \\ \dot{n} &= v_x \sin(\mu) + v_y \cos(\mu) \\ \dot{\mu} &= r - \dot{s}\kappa(s) \end{aligned} \tag{12}$$

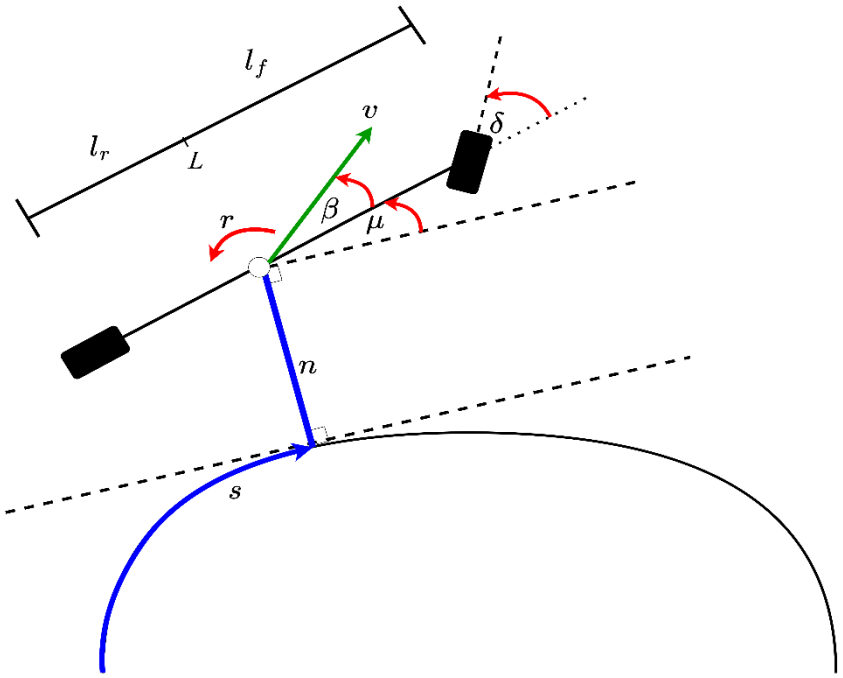

Fig. 5. Illustration of the kinematic bicycle model in the curvilinear co-ordinate system

There are several intermediate values in the dynamic equations in (12), specifically v_x, v_y and r. Similarly, to the dynamic model these values represent the longitudinal velocity, the latitudinal velocity, and the yaw rate of the car. However, unlike the dynamic model, these values are calculated using the geometric relationships within the model as follows.

$$v_x = cos(\beta)$$

$$v_y = sin(\beta) \tag{13}$$

where β is the slip angle which describes the angle between the direction of the velocity at the centre of gravity of the car and the orientation of the vehicle as is calculated by:

$$\beta = tan^{-1}\left(\frac{l_r\, tan(\delta)}{L}\right) \tag{14}$$

The yaw rate r is calculated using the slip angle, steering angle and velocity as shown by:

$$r = \frac{v}{L} tan(\delta)\, cos(\beta)$$

The states of the kinematic model are denoted by $\overline{\boldsymbol{x}}_{Kin}$ and the dynamics described in (12) as $\dot{\boldsymbol{x}_{Kin}} = g_t(\boldsymbol{x}_{Kin}, \boldsymbol{u}_{Kin})$.

6.3 Local Planner

When a collision with an obstacle is detected, a short-term plan is calculated using a local planner. This planner is similar to the global planner, using the same track model, but the simpler kinematic bicycle model is used instead of the dynamic model. However, only a small subsection of the track is used. Specifically, the section spanning from the current location of the vehicle to a planning horizon distance, determined both by the range of the LiDAR sensor that is used and the computational power available on the car. This horizon track is then discretized with a discretization distance of Δs such that there are N_L discretized points, where $\Delta s(N+1)$ is equal to the total planning horizon distance. Additionally, the state dynamics in (12) are transformed into space-dependent equations using the same method shown in (3) and discretized identically to (7) creating the discrete space-based state dynamics for the kinematic model:

$$\begin{aligned} \boldsymbol{x}_{Kin_{j+1}} &= g_s^d\left(\boldsymbol{x}_{Kin_j}, \boldsymbol{u}_{Kin_j}\right) \\ &= \boldsymbol{x}_{Kin_j} + \Delta s\; g_s\left(\boldsymbol{x}_{Kin_j}, \boldsymbol{u}_{Kin_j}\right) \end{aligned} \tag{15}$$

Additional constraints are added to the optimization problem to make it suitable for the local path planning process. This includes constraining the initial state in the local plan to match the current state of the car $\boldsymbol{X}(s_{current})$. In order to avoid the obstacles in the track, we include a constraint to ensure that the car does not collide an obstacle as shown in (12).

$$\begin{aligned} n + \frac{L_c}{2}\sin(|\mu|) + \frac{W_C}{2}\cos(\mu) &\geq O_L(s) \\ n + \frac{L_c}{2}\sin(|\mu|) + \frac{W_C}{2}\cos(\mu) &\leq -O_R(s) \end{aligned} \tag{16}$$

The constraint in (16) uses the left width of the object $O_L(s)$ and the right width of the $O_R(s)$, and ensures that the car cannot enter the obstacle's space. The constraint in (12) is formulated for every obstacle that is detected. We denote this set of constraints as $\boldsymbol{x}_{Kin} \in X_{free}$.
Additionally, we introduce a constraint that limits the lateral acceleration of the car, in order to reduce the inaccuracy of the kinematic model.[4]

$$\frac{v^2}{R} \leq C \tag{17}$$

where R is the radius of curvature and C is a precalculated constant that depends on the friction curve associated with the car tyre and the surface of the road. We denote this constraint as $\boldsymbol{u}_{Kin} \in U_{accel}$.

Finally, we add an additional term to the which penalises the deviation of the car from the global path, as shown in (18).

$$P(\boldsymbol{x}_{Kin_k}) = \left(n_k - n_{Global\ plan}\right)^2 \tag{18}$$

where $n_{Global\ plan}$ is the value for the state n in the global plan at that specific point on the track s.

Therefore, the complete local path planning problem is given by:

$$\begin{aligned} \min_{\boldsymbol{x}_{Kin}, \boldsymbol{U}_{Kin}} & \sum_{j=0}^{N_L} \frac{\Delta s}{\dot{s}_j} + B\left(\boldsymbol{x}_{Kin_j}\right) + P(\boldsymbol{x}_{Kin_k}) \\ s.t. \quad & \boldsymbol{x}_{Kin_{j+1}} = g_s^d\left(\boldsymbol{x}_{Kin_j}, \boldsymbol{u}_{Kin_j}\right) \\ & \boldsymbol{x}_{Kin_0} = \boldsymbol{X}(s_{current}) \\ & \boldsymbol{x}_{Kin} \in X_{track} \\ & \boldsymbol{x}_{Kin} \in X_{friction} \\ & \boldsymbol{x}_{Kin} \in X_{free} \\ & \boldsymbol{u}_{Kin} \in U_{accel} \\ & j = 0, \dots, N_L \end{aligned} \tag{19}$$

However, every such obstacle constraint in (12) introduces a discontinuity in the continuous range of values that the state n can take on. Formulating and solving an optimization problem that contains such a discontinuity is not feasible in real-time. Therefore, when we create a local plan, we do not use the full width of the track and include the discontinuities, instead we limit the allowable width the car to one of the single continuous intervals that make up the full allowable range of n at each discretized point on the track. The number of different continuous intervals that make up the allowable space that the car may enter at any point s is $(U+1)$ where U is the number of obstacles that have been identified. Therefore, in order to determine the optimal path for the horizon distance, it would be necessary to formulate and solve $(U+1)$ optimisation problems and select the best result. This is computationally expensive however and would make formulating a solution in real-time challenging.

Therefore, we create $(U+1)$ subsets that represent the constraint $\boldsymbol{x}_{Kin} \in X_{free}$. Each set represents a continuous interval for range of values that the state n may be at each point along the centre line of the track. We denote each set as $\boldsymbol{x}_{Kin} \in X_{channel_i}$ where i can range from 0 to U. These sets are not necessarily mutually exclusive, but the union, is equivalent to the original $\boldsymbol{x}_{Kin} \in X_{free}$. These sets are then ordered by the minimum range of the state n in each set. If the minimum range of values for n is not larger than the width of the car, then the set is discarded. An optimization problem is then formulated and solved for the set with the largest minimum range for n. We denote this set as the car as $\boldsymbol{x}_{Kin} \in X_{max\ channel}$.
Thus, the local path planning algorithm is given by:

$$\min_{x_{Kin},U_{Kin}} \sum_{j=0}^{N_L} \frac{\Delta s}{\dot{s}_j} + B\left(\boldsymbol{x}_{Kin_j}\right) + P\left(\boldsymbol{x}_{Kin_k}\right)$$
$$s.t. \quad \boldsymbol{x}_{Kin_{j+1}} = g_s^d\left(\boldsymbol{x}_{Kin_j}, \boldsymbol{u}_{Kin_j}\right)$$
$$\boldsymbol{x}_{Kin_0} = \boldsymbol{X}(s_{current})$$
$$\boldsymbol{x}_{Kin} \in X_{track}$$
$$\boldsymbol{x}_{Kin} \in X_{friction}$$
$$\boldsymbol{x}_{Kin} \in X_{max\,channel}$$
$$\boldsymbol{u}_{Kin} \in U_{accel}$$
$$j = 0, \dots, N_L \tag{20}$$

This optimisation problem is then formulated and solved in a similar way as the global plan and the solution is executed by the car, before continuing to execute the global plan.

7 Results

The system was tested using the F1Tenth simulation environment [21]. To test the output of the global path planner, we use a pre-mapped racetrack with a known centre line and run the global path planning algorithm with varying constraint on the maximum tyre force that is permissible. We do this by modifying the value of lamda in (6) which is directly proportional to the maximum tyre force permissible for each tyre. The initial states are set such that the car starts in line with the track, on the centre line with an initial longitudinal velocity of 1 m/s. The output path from the global planner can be seen in figure 5. There is very little difference in the planned path for each different maximum tyre force. However, when examining the velocity values that are given by the path planning algorithm, as plotted in figure 6 and 7, we can see that the maximum permissible tyre forces have a large impact on the speed at which the car can traverse the track. The lap times of each different simulation

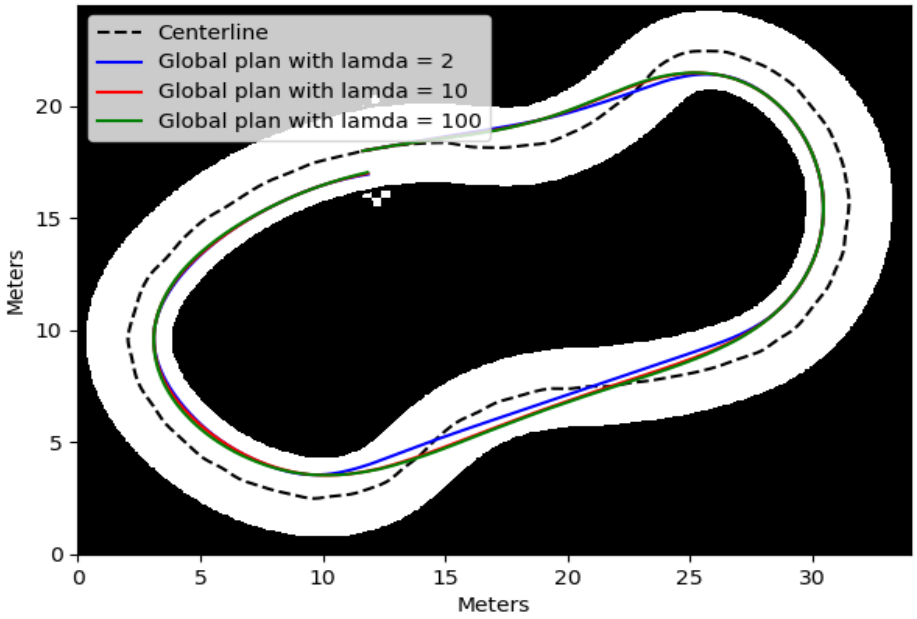

Fig. 6. Minimum lap-times paths generated by the global path planner.

is shown in table 2. These results show if more force that is allowed to act on the tyres, the vehicle can accelerate to a greater degree, and maintain a higher speed, therefore allowing the car to complete the lap in less time.

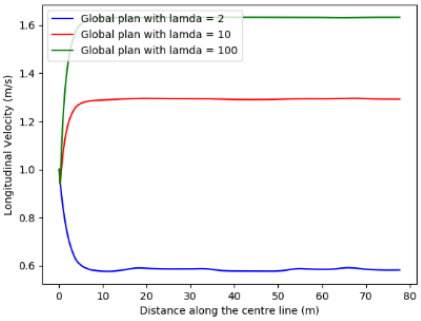

Fig. 8. The longitudinal velocity of the car along the centre line of the track

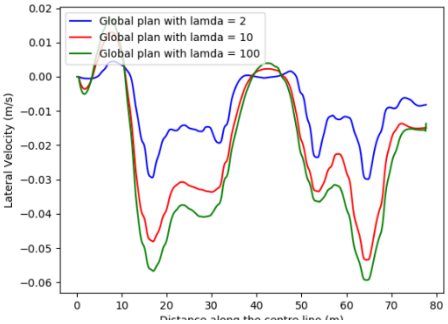

Fig. 7. The latitudinal velocity of the car along the centre line of the track

Table 1: Lap times for global plans formulated with different lambda values

Lamda Value	Lap Time (s)
2	28.55
10	18.23
100	16.96

In order to test the collision avoidance of the autonomous system, we placed an unexpected obstacle in the path of the vehicle while it was executing the global path. First the global plan for the track is created, and then the car is placed at the beginning of the track and begins to execute that plan. Then, when the collision predictor alerts the car of the impending obstacle, the local planner created a short-term plan as seen in figure 9. Figure 10 shows the car then executing the short-term plan, while figure 11 shows that the car re-joins the global plan once it has successfully executed the local plan.

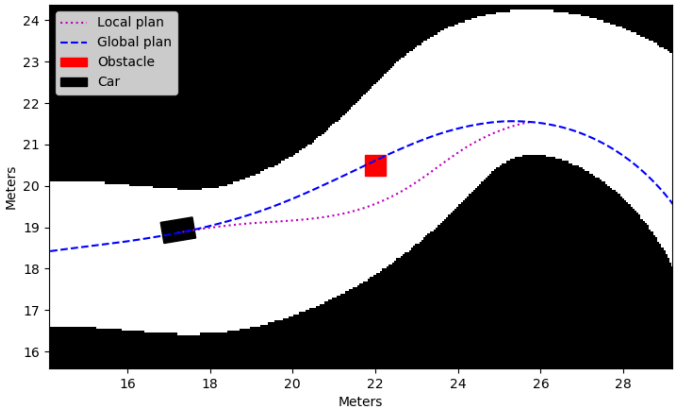

Fig. 9. The global path and the short-term plan generated by the local planner.

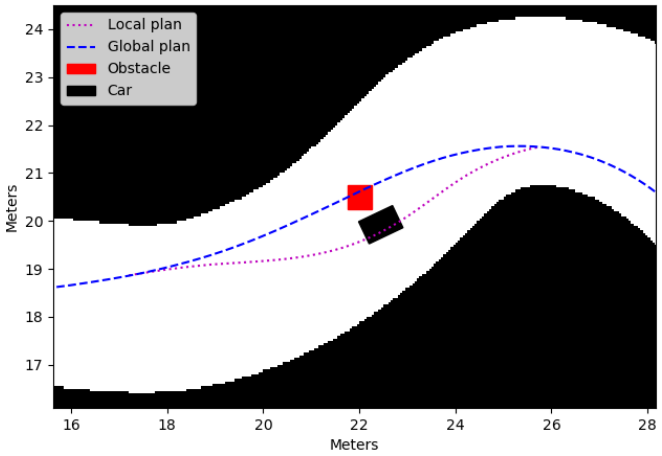

Fig. 10. The car executing the short term-plan.

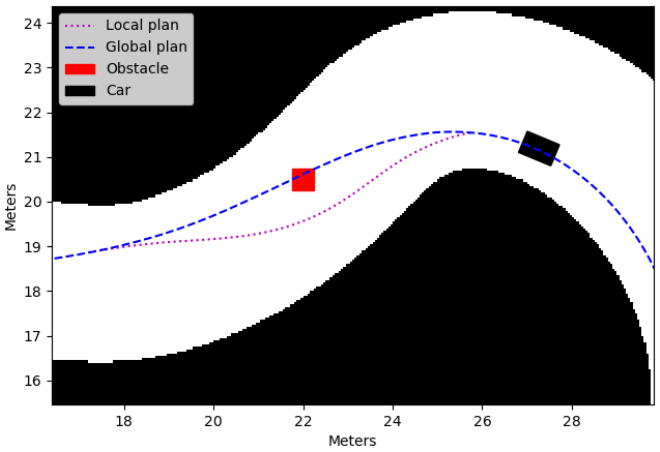

Fig. 11. The car successfully re-joining the global path.

8 Conclusion

The simulation results show that an autonomous vehicle using the hierarchical motion planner is able to race around the track with minimum lap time while avoiding unexpected obstacles.

The long-term motion planner successfully generates the minimum lap-time path while accounting for the dynamics of the vehicle. The motion plan is successfully executed by the car without any additional expensive computations. This is demonstrated by the different lap times for each motion plan generated with different vehicle dynamics and parameters.

The obstacle avoidance capabilities of the autonomous racing system have also been proven successful. When an unexpected obstacle is placed in the track, it is successfully detected by the autonomous racing system. When a collision is predicted by the autonomous racing system, the local planner is able to generate a short-term plan that successfully avoids the obstacle, and re-joins the global path, while also accounting for the vehicle dynamics and continuing to race competitively. Therefore, the hierarchical motion planner proposed in this paper is a viable solution to computationally inexpensive for autonomous racing.

In future work, the collision avoidance system will be tested to see if it can be successfully expanded to include dynamic obstacles as well as implemented in practical applications.

References

1 https://www.formulastudent.de/about/concept/
2 **https://f1tenth.org/build**
3 A. Liniger, A. Domahidi, M Morari, *Optimization-Based Autonomous Racing of 1:43 Scale RC Cars*. Optimal Control Applications and Methods, vol **36**, 628-647. (2015)
4 P. Polack, F. Altché, B. Novel, A. de La Fortelle, *The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?*, in Proceedings of the IEEE Intelligent Vehicles Symposium, IEEE, 812-818. (2017)
5 V. Fors, B. Olofsson, L. Nielsen, *Formulation and interpretation of optimal braking and steering patterns towards autonomous safety-critical manoeuvres*. Vehicle System Dynamics, vol **57**, 1-1, (2018)
6 T. Novi, A. Liniger, R. Capitani, C. Annicchiarico, *Real-time control for at-limit handling driving on a predefined path*. Vehicle System Dynamics. vol **58**. 1-30. (2019)
7 J. Kong, M. Pfeiffer, G. Schildbach, F. Borrelli, *Kinematic and dynamic vehicle models for autonomous driving control design*, in Proceedings of the IEEE Intelligent Vehicles Symposium, IEEE, 1094-1099. (2015)
8 M. Ahlberg, *Optimization Based Trajectory Planning for Autonomous Racing*, Masters Thesis (2018)
9 C. Samak, T. Samak, S. Kandhasamy, *Autonomous Racing using a Hybrid Imitation-Reinforcement Learning Architecture*, arXiveprints (2021)
10 A. Heilmeier, A. Wischnewski, L. Hermansdorfer, J. Betz, M. Lienkamp, B. Lohmann, *Minimum curvature trajectory planning and control for an autonomous race car.* Vehicle System Dynamics, vol **58**. 1-31, (2020)
11 José L. Vázquez and Marius Brühlmeier and Alexander Liniger and Alisa Rupenyan and John Lygeros. *Optimization-Based Hierarchical Motion Planning for Autonomous Racing*, in Proceedings of the IEEE International Conference on Intelligent Robots and Systems, IEEE, 2397-2403. (2020)
12 T. Sjolin and A. Sundberg, *Trajectory planning and control of an autonomous race vehicle*, Masters Thesis, 2021.
13 A. Rucco, G. Notarstefano, J. Hauser, *An Efficient Minimum-Time Trajectory Generation Strategy for Two-Track Car Vehicles*, IEEE Transactions on Control Systems Technology. vol **23**. (2015)
14 Liniger, A. *Path Planning and Control for Autonomous Racing*, Doctoral Thesis (2018)
15 H.B. Pacejka and E. Bakker, *The magic formula tyre model*, Vehicle system dynamics, vol **21**, pp.1–18. (1992)
16 M.L. Bynum, G. Hackebeil, W. Hart, C. Laird, B. Nicholson, J. Siirola, J. Watson, D. Woodruff. *Pyomo - Optimization Modeling in Python*. Third Edition Vol. **67**. Springer (2021)
17 W, Hart, J. Watson, D. Woodruff, *Pyomo: modeling and solving mathematical programs in Python*, Mathematical Programming Computation, vol **3,** 219-260. (2011)
18 A. W ̈achter, L. T. Biegle, *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming*, Mathematical programming, vol **106**, 25–57 (2006)
19 K. Vivek, M.A Sheta, V. Gumtapure, *A Comparative Study of Stanley, LQR and MPC Controllers for Path Tracking Application*, in Proceedings of the IEEE International Conference on Intelligent Systems and Green Technology, 67-674. (2019)

20 M. Ester, H.P. Kriegel, J. Sander, X. Xu et al. *A density-based algorithm for discovering clusters in large spatial databases with noise*, in Proceedings of the International Conference on Knowledge Discovery and Data Mining ,vol. **96**, 226–231 (1996)
21 M. O'Kelly, et al., *F1TENTH: An Open-source Evaluation Environment for Continuous Control and Reinforcement Learning*, in Proceedings of the NeurIPS 2019 Competition and Demonstration Track, 77-89. (2020)