Obstacle Aware Planning on BEVFormer Generated Environments with NMPC-CBF

ROB 535 Project Report

Chitra Devulapalli, Peter Redman, Pannaga Sudarshan

***Abstract*— This work develops an NMPC-CBF based planner for autonomous vehicles, leveraging the outputs from BEVFormer– a state-of-the-art framework for generating Bird's Eye View representations. GitHub: https://github.com/PannagaS/rob-535-final-project**

I. Introduction

Autonomous driving systems need robust planners to navigate dynamic environments while ensuring safety constraints. The perception-to-control pipeline, where environmental understanding influences trajectory planning, is crucial to achieve good performance.

In this work, we propose a Nonlinear Model Predictive Control (NMPC) planner integrated with Control Barrier Functions (CBFs) to build a robust and adaptive path-planning framework that leverages BEVFormer [1] outputs. The NMPC framework ensures the generation of optimal trajectory in real time by predicting vehicle dynamics over a finite horizon, while CBFs enforce safety-critical constraints, such as collision avoidance and adherence to the road boundary. Using the nuScenes v1.0-mini dataset [2], we demonstrate the effectiveness of our approach in generating safe and efficient trajectories in complex environments.

This integration bridges perception and control, showcasing how BEV representations can directly inform advanced planning frameworks. The NMPC-CBF planner [3] adapts dynamically to environmental changes, offering a reliable solution for autonomous driving systems.

II. Methodology

A. BEVFormer

The BEVFormer takes in a set of video feeds (and thus a sequence of sets of images) collect from a multi-camera array atop a vehicle, and produces a single two-dimensional image of the scene as viewed from above. Figure 1 shows and example BEVFormer output.

B. Image Processing

Since the BEVFormer generates only an image as output, rather than any explicit representation of the locations of the ego vehicle and relevant obstacles, we used conventional image processing techniques to extract this information from the resultant image. We developed separate but highly similar algorithms for the tasks of (1) extracting the location of the ego vehicle and (2) extracting the location, size, and orientation of relevant obstacles. All of our image processing was performed using OpenCV [4].

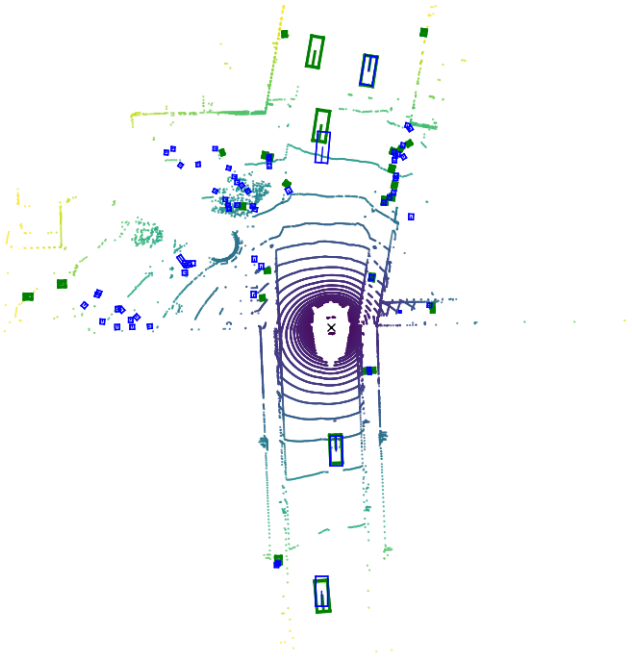

Fig. 1. Example BEVFormer output. Objects drawn in green are ground truth representations, while objects drawn in blue are the predicted representations from the BEVFormer. The position of the ego vehicle is marked with a black 'x', visible here at the center of the LIDAR scans.

To extract the ego vehicle location, we first convert the input image to grayscale and then threshold the grayscale image. We then use `cv2.findContours` to find all contours in the thresholded image and identify the largest of them. The center coordinates of this largest contour are computed and returned.

To extract obstacle locations and characteristics, we first apply an HSV color filter to isolate all blue pixels in the input image. We then convert to grayscale, threshold, and identify all contours in the result. For each contour with area above a threshold, we compute its center, size, and orientation using the `cv2.minAreaRect` function. We package all of these results into `NumPy` arrays and return them. The center coordinates and obstacle dimensions are in pixels, and the orientations are in degrees.

C. Coordinate Transformations

The locations and characteristics of the obstacles generated in the previous step are not suitable to be used in our NMPC-CBF problem yet. Primarily, this is because the

system dynamics for our vehicle are defined in SI units (i.e. meters and radians, not pixels and degrees). Secondly, the coordinate frame used by OpenCV would be unintuitive for our problem, so we chose to define our world frame at the ego vehicle center, as shown below in figure 2.

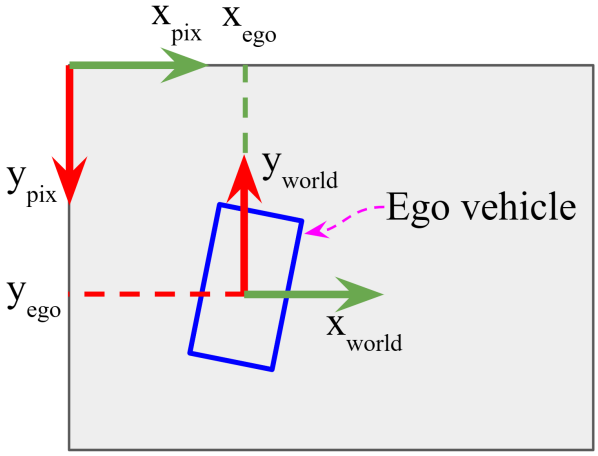

Fig. 2. Figure depicting how we defined our 'world' coordinate frame at the center of the ego vehicle $p_{ego} \triangleq [x_{ego}, y_{ego}]^T$, with its x-axis pointing to the right of the image and y-axis pointing to the top of the image.

The coordinate transformations used are standard homogeneous transformations for two-dimensional coordinates, with the addition of a scaling matrix, and so we will not go into great detail on the derivations of the expressions for the transformation matrices in this paper. Let us first define a scaling matrix S, given a parameter ρ, the number of pixels per meter in the input image.

$$S(\rho) = \begin{bmatrix} \rho & 0 & 0 \\ 0 & \rho & 0 \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3}$$

For our work, we estimated ρ manually, and found a reasonable value to be $\rho = 7.5$. Now, let us define two homogeneous transformation matrices, H_w^p, which transforms coordinates in the scaled world frame to the pixel frame, and H_p^w, which transforms coordinates in the pixel frame to the scaled world frame. Both depend on the location of the origin of the world frame p_{ego}.

$$H_w^p(p_{ego}) = \begin{bmatrix} 1 & 0 & x_{ego} \\ 0 & -1 & y_{ego} \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3}$$

$$H_p^w(p_{ego}) = \begin{bmatrix} 1 & 0 & -x_{ego} \\ 0 & -1 & y_{ego} \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3}$$

We can combine these homogeneous transformation matrices with the scaling matrix S to compute our final transformations $T_w^p(\rho, p_{ego})$ and $T_p^w(\rho, p_{ego})$, which map from world frame to pixel frame, and pixel frame to world frame, respectively

$$T_w^p(\rho, p_{ego}) = H_w^p(p_{ego})S(\rho)$$
$$T_p^w(\rho, p_{ego}) = S^{-1}(\rho)H_p^w(p_{ego})$$

D. Obstacle Representation

From the image processing step, we have a set of M rectangular obstacles in the pixel frame O_{pix}:

$$O_{pix} = \{o_1, ..., o_M\}, \qquad o_i = (c_i, s_i, r_i)$$

where $c_i \in \mathbb{R}^2$ is the center coordinate in pixels, $s_i \in \mathbb{R}^2$ is the dimension of the obstacle in pixels, $r_i \in \mathbb{R}^2$ is the orientation of the obstacle in degrees. To get O_{world} we apply the following transformations:

$$\hat{c}_i^{world} = T_p^w \hat{c}_i$$
$$s_i^{world} = s_i/\rho$$
$$r_i^{world} = -\frac{\pi}{180} r_i$$

Where the hat operator simply denotes the homogeneous form of the center. Finally, we must be able to easily define control barrier functions based on the obstacles, so we will compute a bounding ellipse for each of the rectangular obstacles. We use the general form of a conic section to represent our ellipses:

$$A_i x^2 + B_i y^2 + C_i xy + D_i x + E_i y + F_i = 0$$

Where A_i, B_i, C_i, D_i, E_i, and F_i depend on the parameters of the obstacle $o_i^{world} = (c, s, r)$.

$$A_i = \left(\frac{\cos r}{s_x}\right)^2 + \left(\frac{\sin r}{s_y}\right)^2$$
$$B_i = \left(\frac{\cos r}{s_y}\right)^2 + \left(\frac{\sin r}{s_x}\right)^2$$
$$C_i = \sin(2r)\left(\frac{1}{s_x^2} - \frac{1}{s_y^2}\right)$$
$$D_i = -c_y C - 2c_x A$$
$$E_i = -c_x C - 2c_y B$$
$$F_i = c_x^2 A + c_y^2 B + c_x c_y C - 1$$

These parameter values define the smallest ellipse which both encompasses the obstacle, and shares the same aspect ratio as the obstacle. Of course, the size of the obstacle can be scaled for increased safety, and we chose to scale the sizes of all detected obstacles by a factor of two before defining their bounding ellipses.

E. NMPC-CBF

We used the algorithmic differentiation package Casadi [5] to write our NMPC implementation. Our dynamic model, cost functions, and constraints are detailed in the following sub-sections. At every step of our simulation loop, we perform the following optimization:

$$\min_{\underline{u}\in U} \quad J_{term} + \sum_{k=0}^{N} J_{stage}(k), \qquad \underline{u} = [a, \delta]^T$$

subject to dynamic model constraints, point-wise constraints on control effort and states, and obstacle avoidance constraints.

1) Stage and Terminal Cost Function: The stage cost function penalizes deviations from desired behavior during each stage of the planning horizon.

$$J_{\text{stage}} = W_\psi(\psi_{\text{des}} - \psi_k)^2 + W_d\|d_k\|^2 + W_a a_k^2 + W_\delta \delta_k^2 \quad (1)$$

where $\psi_{\text{des}} = \tan^{-1}(y_g, x_g)$ is the desired heading angle, calculated using the target position $\underline{x}_{\text{des}} = [x_g, y_g]^T$. The vector d is the deviation from the straight-line path to the goal, defined as:

$$d = \underline{x}_k - \frac{\underline{x}_{\text{des}}^T x_k}{\|\underline{x}_{\text{des}}\|^2} \cdot \underline{x}_{\text{des}} \in \mathbb{R}^2 \quad (2)$$

a and δ are the control inputs representing acceleration and steering angle, respectively. The weights $W_\psi = 10$, $W_d = 10$, $W_a = 1$, $W_\delta = 1$ allow tuning of the penalties for heading deviation, positional deviation, acceleration effort, and steering effort, respectively.

The terminal cost function penalizes the final deviation from the desired goal state, encouraging the vehicle to converge to the target position by the end of the planning horizon. Terminal cost is as defined in equation 3.

$$J_{\text{term}} = W_x(x_g - x_N)^2 + W_y(y_g - y_N)^2 \quad (3)$$

Here, (x_N, y_N) represent the final state of the ego-vehicle after planning the horizon, and $\underline{x}_{\text{des}} = [x_g, y_g]^T$ is the target position. $W_x = 1000$ and $W_y = 1000$ are weights assigned to x and y position deviations.

2) Dynamic Model: The dynamic model is given in equation 4.

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ \psi \\ v \end{bmatrix} = \begin{bmatrix} v\cos(\psi + \beta) \\ v\sin(\psi + \beta) \\ \frac{v}{L_r}\sin\beta \\ a \end{bmatrix} \quad (4)$$

where $\beta := \tan^{-1}(\frac{L_r}{L_r + L_f}\tan^{-1}\delta)$. States x, y, ψ, and v are ego-vehicle's x, y position, heading angle, and velocity respectively. L_r and L_f are distances from rear and front axles to the center of mass of the ego-vehicle respectively, and a is the acceleration of the ego-vehicle. It is assumed that the dynamic model follows SI units.

3) Constraints: Our NMPC formulation has three sets of constraints namely, dynamics constraints, state constraints, and the initial state constraint. The dynamics constraints ensure that the vehicle follows the dynamic model:

$$\underline{x}_{k+1} = f(\underline{x}_k, \underline{u}_k) \qquad k = 0, ..., N$$

and the initial state constraint ensures that the MPC process starts from the current location of the vehicle:

$$\underline{x}_0 = \underline{x}_k$$

The state constraints limit the steering rate and maximum lateral acceleration of the vehicle and enforce collision avoidance using control barrier functions. For an obstacle ellipse parameterized as (A, B, C, D, E, F), we define:

$$h_k = Ax_k^2 + By_k^2 + Cx_k y_k + Dx_k + Ey_k + F$$

And enforce, for $0 \leq \gamma \leq 1$:

$$h_{k+1} \geq (1 - \gamma)h_k \quad (5)$$

For our experiments, we found $\gamma = 0.15$ to be a value which produced good results.

III. Results

BEVFormer was implemented on nuScenes v1.0-mini dataset, and the corresponding surround view detections and bird's eye-view are shown in Figure 3 and Figure 4 respectively.

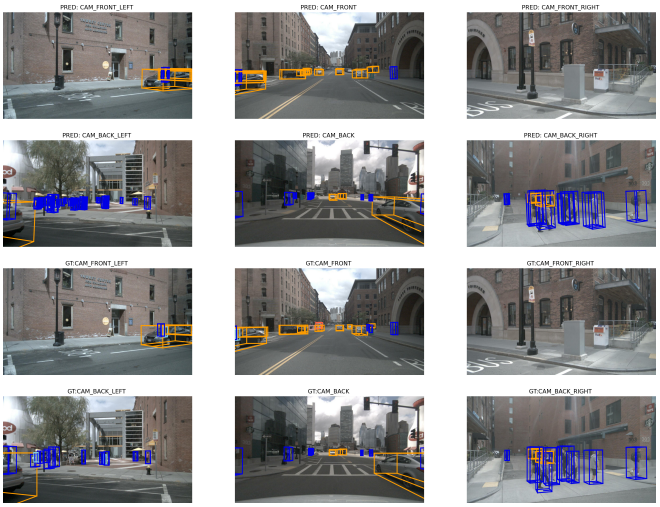

Fig. 3. 3D boxes predictions in multi-camera images and the bird's-eye-view.

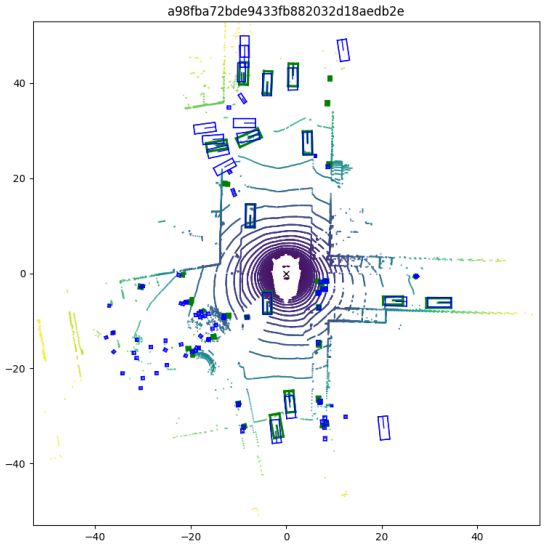

Fig. 4. Bird's Eye View of the ego-vehicle.

Figure 5 shows the trajectory of the ego vehicle planned after executing NMPC controller with CBF for safety guar-

antees, starting from $(0, 0)$ (origin of the ego vehicle) to $(4.7, 45.6)$ m which is our goal location.

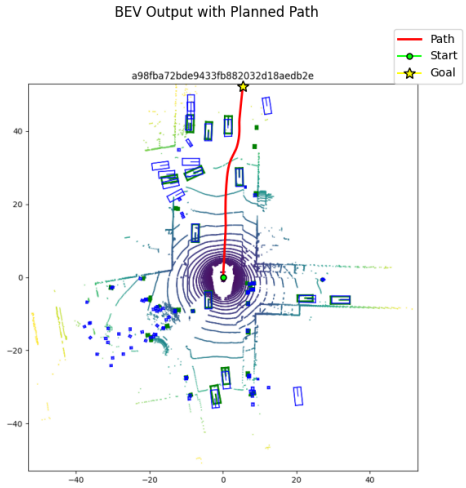

Fig. 5. Planned trajectory of the ego vehicle.

Figure 6 illustrates the barriers represented as ellipsoids encapsulating each obstacle predicted by BEVFormer. Ellipsoids are better suited for representing constraints that arise from the relative motion of obstacles and the ego-vehicle. In this scenario, we treat every vehicle other than ego-vehicle as our obstacle. This alignment allows the CBF to enforce constraints more effectively without overly restricting feasible paths. The hyperparameter γ is set to 0.15 in equation 5 in our implementation, which is subjective to the chosen goal location and upper and lower bounds on control input.

The ego-vehicle carefully maneuvers around the obstacles seamlessly while still satisfying the state and control constraints, and the dynamic model. The integration of CBF with the NMPC controller guarantees that the ego-vehicle remains within the operational limits, preventing abrupt maneuvers by considering safety constraints continuously over the planning horizon.

Further, we observe the evolution of ego-vehicle's state and control in Figure 7. We note that the ego-vehicle takes ≈ 17 seconds to reach the goal location.

IV. Future Work

The current implementation of CBF with an NMPC controller for trajectory planning is done assuming that the hyperparameter γ is fixed. This strategy can be either too conservative (over-constraining the system) or too risky (allowing unsafe behaviors) for various scenarios. A better way would perhaps be to dynamically set γ, given the system environment and its current state. This makes the system more robust to changes in the environment, and ensuring a better balance between performance and safety.

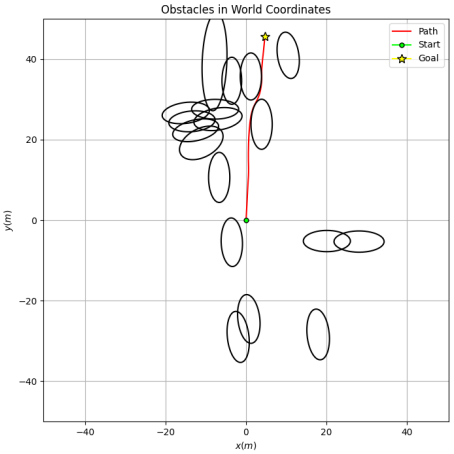

Fig. 6. Obstacles visualized as ellipsoids.

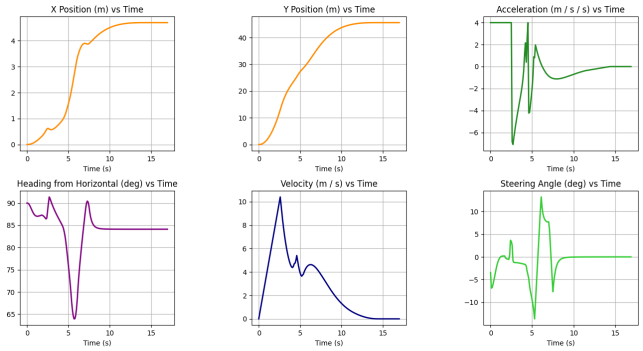

Fig. 7. state and control signal envolution over time.

A high γ can be employed in risky situations (when the ego-vehicle is in close proximity to its obstacles), and γ can be more relaxed when cruising through open-space. In addition, increasing the planning horizon of the NMPC controller can potentially anticipate the effects of actions further into the future, potentially leading to smoother and more optimal trajectories.

V. Conclusion

In this work, we successfully implemented BEVFormer, a novel approach for generating Bird's Eye View (BEV) representations of the ego-vehicle and its surroundings from surround-camera images. We further this pipeline by converting this BEV to an occupancy grid while carefully preserving all the critical details to enable Nonlinear Model Predictive Control (NMPC) for trajectory optimization. The system plans a path from point A to point B, where point A is the ego-vehicle's current location and point B is any arbitrary goal location while ensuring safety through the integration of a Control Barrier Function (CBF). Our pipeline was effectively realized and demonstrated using the nuScenes v1.0-mini dataset, with results thoroughly analyzed and reported.

REFERENCES

[1] Z. Li, W. Wang, H. Li, E. Xie, C. Sima, T. Lu, Q. Yu, and J. Dai, "Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers," 2022. [Online]. Available: https://arxiv.org/abs/2203.17270

[2] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," *arXiv preprint arXiv:1903.11027*, 2019.

[3] J. Zeng, B. Zhang, and K. Sreenath, "Safety-critical model predictive control with discrete-time control barrier function," 2021. [Online]. Available: https://arxiv.org/abs/2007.11718

[4] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[5] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.