Navigating in Dynamic Environments via Simultaneous Geometry-Aware Prediction and Planning

Mikolaj Kliniewski[1,⋆], Arnaud Bersier[2,⋆,†], Jesse Morris[1,†], Yiduo Wang[1], Stefan Leutenegger[2] and Viorela Ila[1]

***Abstract*— Accurate estimation and prediction of object dynamics can fundamentally transform robot navigation by enabling controllers to reason about future scene evolution rather than merely reacting to the present. This work integrates accurate dynamic object prediction derived directly from state estimation and incremental dense reconstruction of moving obstacles into two navigation controllers: a factor-graph-based MPC specifically designed for dynamic environments and a Nav2 MPPI controller extended through a custom critic. The MPC simultaneously optimizes object trajectory prediction and robot planning, while the MPPI critic performs predictive, geometry-aware collision evaluation using time-projected per-object ESDF submaps. Both approaches are evaluated in a Gazebo warehouse simulation with Husarion RosBot XL platforms, demonstrating improved navigation performance in dynamic and geometrically complex environments.**

I. Introduction

Real-world environments are highly dynamic and change rapidly. In such conditions, safe and efficient navigation remains a fundamental challenge for mobile robots. Recent advances in dynamic Simultaneous Localization and Mapping (SLAM) [1]–[3] produce high-quality per-object motion and camera pose estimates, while latest volumetric mapping techniques [4] provide representations capable of reflecting scene dynamics. Nevertheless, most navigation algorithms do not leverage such understanding, remaining reactive rather than anticipating the trajectory and geometry of obstacles.

While recent progress in predicting obstacle trajectories at planning time has shown promising results [5]–[7], simultaneous trajectory prediction and planning within a single optimization framework remains an open problem. Additionally, object geometry remains under-utilized in collision avoidance: obstacles are conventionally simplified into geometric primitives, despite modern perception techniques able to reconstruct dynamic objects incrementally online.

To address these limitations and problems, we propose two approaches that adopt different design choices for modeling obstacles in dynamic environments. First, we propose a factor-graph-based Model Predictive Control (MPC) to represent objects as pose-and-sphere models and simultaneously optimize objects' trajectory predictions and ego-robot planning. Second, we incorporate per-object ESDF submaps into trajectory scoring within a custom Nav2 MPPI critic, projecting object geometry forward in time for geometry-aware collision evaluation.

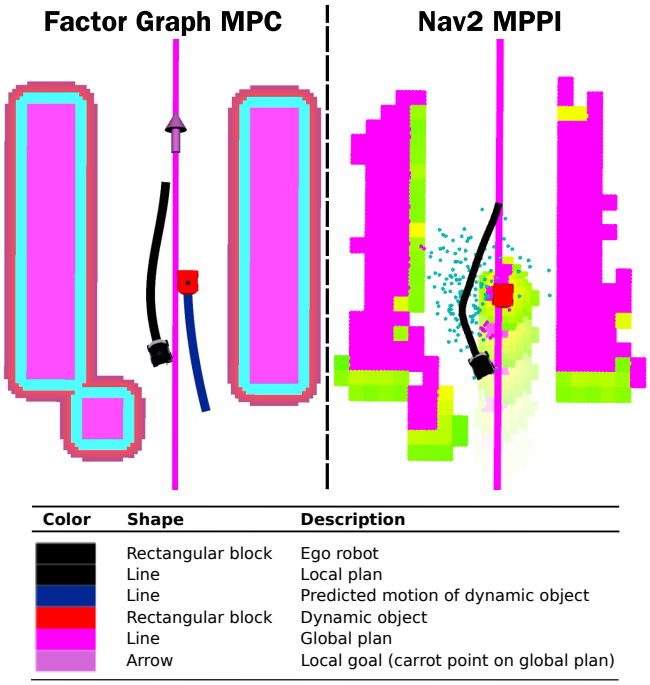

Fig. 1: Ego robot navigating around a moving obstacle in a narrow corridor, addressed by both proposed approaches: the factor graph MPC and the Nav2 MPPI with a custom dynamic obstacle critic.

II. Related Work

This section reviews the two most relevant topics for predictive navigation in dynamic environments, namely factor-graph-based MPC for simultaneous planning and prediction, and geometry-aware navigation using dense obstacle reconstruction.

Predicting obstacle motion is critical for safe and efficient navigation in dynamic environments, as purely reactive planners cannot anticipate future interactions and may produce overly conservative or unsafe behaviors [8], [9]. Among optimization-based prediction methods, factor-graph-based MPC has been well established for both static [10]–[13] and dynamic environments. Table I summarizes the extension of MPC to dynamic settings, highlighting object representations and whether trajectory prediction can be updated collaboratively with the ego robot plan.

Predicting obstacle motion has been shown to significantly improve trajectory safety and quality compared to purely

[1] Mikolaj Kliniewski, Jesse Morris, Yiduo Wang and Viorela Ila are with the Australian Centre for Robotics (ACFR), University of Sydney, Sydney, Australia. mikolaj.kliniewski, jesse.morris, yiduo.wang, viorela.ila@sydney.edu.au

[2] Arnaud Bersier and Stefan Leutenegger are with the Mobile Robotics Lab, ETH Zurich, Switzerland. abersier, lestefan@ethz.ch

[†] Work completed during internship.

[⋆] These authors contributed equally to this work.

TABLE I: Factor-graph-based works on planning in dynamic environments. Vars: Object Variables. †: Treats dynamic objects as static obstacles. ⋆: Assumes the objects' future trajectories are known.

Method	Approximation	Representation	Collaborative
Reactive SCATE [7]	sphere	SDF †	✗
Predictive SCATE [7]	sphere	SDF ⋆	✗
RHAP-GPMP2 [14]	cylinder	ESDF	✗
DynoJEPP [18]	sphere	ESDF + Vars	✓

reactive replanning [8], [9]. King-Smith *et al.* [7] proposed Simultaneous Control and Trajectory Estimation (SCATE), including a predictive variant that incorporates known future object trajectories through a time-varying SDF. RHAP-GPMP2 [14] integrates predicted human trajectories into a composite distance field [15] within a factor-graph-based GPMP2 planner [16], [17]. However, both Predictive SCATE and RHAP-GPMP2 treat prediction and planning as separate stages.

DynoJEPP [18] addresses this limitation by jointly estimating ego pose, object motion, object trajectory prediction, and robot plan. Future obstacle motions are treated as optimization variables, enabling mutual interaction between robot and obstacle trajectories. Rather than embedding obstacles in the map, DynoJEPP directly penalizes proximity to predicted object locations through optimization factors, achieving safe and predictive navigation in dynamic environments.

Among sampling-based methods, Model Predictive Path Integral (MPPI) [19], the default local planner in Nav2 [20], approximates the optimal control law by evaluating K Gaussian-perturbed candidate control sequences in parallel, but assumes a static environment in its original formulation.

Expanding on MPPI to incorporate predictions of dynamics, Trevisan *et al.* [5] model probabilistic obstacle predictions as Mixtures of Gaussians, and Mohamed *et al.* [6] reformulate probabilistic collision checking as a deterministic Mahalanobis distance condition. However, these methods discard any information on dynamic obstacle geometry and reduce them to 2D circular footprints, a convention also employed by methods in Table I.

Exploiting obstacle geometry for predictive navigation requires a reconstruction layer that can represent dynamic objects and project their geometry forward in time. However, most dense volumetric reconstruction methods assume static scenes [21], [22], causing dynamic objects to leave artifacts in the map. Methods designed for dynamic environments either focus on static free-space reconstruction while neglecting dynamic objects [23], [24], or reconstruct dynamic objects without modeling free space [25], [26], limiting their applicability to navigation.

DynORecon [4] incrementally reconstructs dynamic objects' geometry and explicitly models known free space, but provides no temporal prediction. Propagating its per-object submaps forward in time using an object motion model enables geometry-aware, temporally-informed trajectory evaluation within a receding-horizon framework such as MPPI.

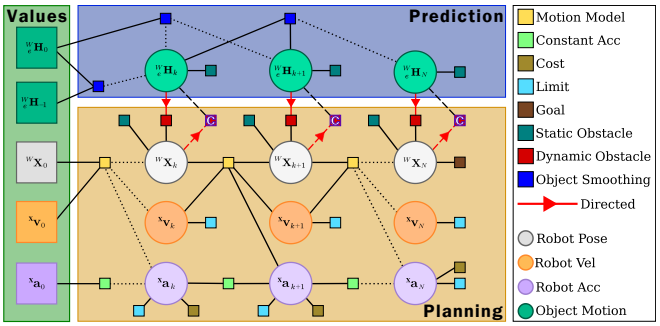

Fig. 2: Factor graph for simultaneous trajectory prediction and planning in dynamic environments. The "Values" (green) are provided by the estimation module or previously executed commands. The "Prediction" module (blue) estimates future motions and poses of objects, while the "Planning" module (orange) computes future ego-robot actions and poses.

III. Factor-Graph-Based MPC

The foundation of the factor-graph-based formulation discussed in this section was originally introduced in DynoJEPP [18]. In this work, we decouple the future-based variables from the past-based ones, focusing on online execution of the simultaneous trajectory prediction and local planning.

We employ a factor graph to estimate the values of state variables θ by optimizing over probabilistic factors ϕ that encode non-linear constraints between variables. Each ϕ is defined by a residual error function $\mathbf{r}(\cdot)$ associated with its covariance Σ. By assuming ϕ_i represents a likelihood over the set of involved variables with Gaussian noise, inference in the factor graph is equivalent to solving a nonlinear least-squares optimization.

A. Notations

Our formulation defines robot poses, velocities and accelerations as:

$$\mathcal{X}=\{{}^{W}\mathbf{X}_k\}_{k\in\mathcal{K}}, \mathcal{V}=\{{}^{X_k}\mathbf{v}_k\}_{k\in\mathcal{K}}, \mathcal{A}=\{{}^{X_k}\mathbf{a}_k\}_{k\in\mathcal{K}}$$

where $\{W\}$ is the world frame and ${}^{W}\mathbf{X}_k \in \mathrm{SE}(2)$ is the robot's pose in $\{W\}$ at discrete planning step k, associated with its body-fixed frame X_k. $\mathcal{K} = \{0, 1, 2, \ldots, N\}$ is the set of steps over the planning horizon N. As velocity and acceleration share the same time step as their reference frames, and the reference frames are consistent throughout the paper, we simplify the notation to ${}^{X_k}\mathbf{v}_k \equiv \mathbf{v}_k, {}^{X_k}\mathbf{a}_k \equiv \mathbf{a}_k$, and the robot pose to ${}^{W}\mathbf{X}_k \equiv \mathbf{X}_k$.

For any moving object, its pose is denoted as ${}^{W}\mathbf{L}_k \in \mathrm{SE}(3)$, while its rigid-body motion is ${}^{W}_{a}\mathbf{H}_b \in \mathrm{SE}(3)$ for arbitrary time step from a to b, following the Dynamic SLAM works [3], [18], [27], [28]. In our MPC formulation, objects are represented as spheres, and the object pose is therefore defined at the center of the sphere. Following the work of [18], we utilize the direct estimation of object motion ${}^{W}_{e}\mathbf{H}_k$ provided by the DynoSAM framework [28] to track the moving objects given the fixed reference pose ${}^{W}\mathbf{L}_e$:

$${}^{W}\mathbf{L}_k = {}^{W}_{e}\mathbf{H}_k \, {}^{W}\mathbf{L}_e. \tag{1}$$

We formulate the problem using factor graphs, where factors involving the object motion variable ${}^{W}_{e}\mathbf{H}_k$ obtain the corresponding object pose ${}^{W}\mathbf{L}_k$ through the constant reference pose ${}^{W}\mathbf{L}_e$ via Eq. (1). Since the reference pose is fixed for each object we simplify the notation to ${}^{W}_{e}\mathbf{H}_k \equiv {}^{W}\mathbf{H}_k$.

B. Trajectory Prediction

The prediction module in Fig. 2 (top) performs scene-aware object trajectory prediction in $\mathrm{SE}(3)$. To ensure realistic future interactions between the robot and dynamic objects, the predicted object states must be synchronized with the planned robot poses. Since the object smoothing factors in Eq. (4) assume a constant time interval between consecutive motion estimates, the elapsed time between the estimated motions ${}^{W}\mathbf{H}_{-1}$ and ${}^{W}\mathbf{H}_0$ must match the MPC planning time step, Δt. These estimated object motions are incorporated into the factor graph as fixed values.

Given the estimated object poses ${}^{W}\mathbf{L}_0$ and ${}^{W}\mathbf{L}_{0-\Delta a}$, corresponding to the current object pose and the pose estimated one estimation interval earlier, respectively, the pose ${}^{W}\mathbf{L}_{-1}$ can be obtained through backward interpolation. Here, Δa denotes the interval between consecutive object estimates and is not necessarily equal to the MPC planning time step Δt:

$$\begin{aligned} {}^{W}\mathbf{L}_{-1} &= {}^{W}\mathbf{L}_0 \exp\left(-\frac{\Delta t}{\Delta a}\log\left({}^{L_{0-\Delta a}}_{0-\Delta a}\mathbf{H}_0\right)\right) \\ {}^{L_{0-\Delta a}}_{0-\Delta a}\mathbf{H}_0 &= \left({}^{W}\mathbf{L}_{0-\Delta a}\right)^{-1}{}^{W}\mathbf{L}_0. \end{aligned} \tag{2}$$

Next, the object CoM constant ${}^{W}\mathbf{L}_e$ is set to ${}^{W}\mathbf{L}_e = {}^{W}\mathbf{L}_{-1}$, and the estimated motions are:

$$ {}^{W}\mathbf{H}_{-1} = \mathbf{I}, \quad {}^{W}\mathbf{H}_0 = {}^{W}\mathbf{L}_0 {}^{W}\mathbf{L}_e^{-1}. \tag{3}$$

Object smoothing factor: The object smoothing factor promotes smooth relative motion change, between consecutive poses, by the dynamic object. The factor is given as:

$$\begin{aligned} &\phi^{\text{smooth}}(\mathbf{H}_{k-2},\mathbf{H}_{k-1},\mathbf{H}_k) \propto \\ &\exp\left\{-\frac{1}{2}\left\|\left(\mathbf{L}_{k-2}^{-1}\mathbf{L}_{k-1}\right)^{-1}\left(\mathbf{L}_{k-1}^{-1}\mathbf{L}_k\right)\right\|^2_{\Sigma_{\text{smooth}}}\right\}. \end{aligned} \tag{4}$$

Static obstacle factor: This factor encourages predicted and planned trajectories to keep any entity (robots and objects) distance from static obstacles. The hinge loss function queries the static SDF map, and penalizes whenever the entity is close to any obstacle. The d_{os} is the obstacle safety distance between the nearest obstacle and the entity's CoM $\mathbf{P}_k$, which can be either the robot pose $\mathbf{X}_k$ or object pose $\mathbf{L}_k$. The hinge loss and factor are defined as:

$$\begin{aligned} &\mathbf{h}_{\text{sdf}}(\mathbf{P}_k) = \begin{cases} d_{\text{os}} - \text{esdf}(\mathbf{P}_k), & \text{esdf}(\mathbf{P}_k) < d_{\text{os}} \\ 0, & \text{otherwise} \end{cases}, \\ &\phi^{\text{s-obs}}(\mathbf{P}_k) \propto \exp\left\{-\frac{1}{2}\|\mathbf{h}_{\text{sdf}}(\mathbf{P}_k)\|^2_{\Sigma_{\text{s-obs}}}\right\}. \end{aligned} \tag{5}$$

C. Local Planning

The planning module in Fig. 2 addresses static-dynamic scene-aware ego robot plans in positional and action space in $\mathrm{SE}(2)$. The current robot pose $\mathbf{X}_0$ is represented in the factor graph as a constant value and comes from the employed estimation source. The current robot velocity $\mathbf{v}_0$ and acceleration $\mathbf{a}_0$ are the most recently executed actions.

Motion model factor: Similarly to DynoJEPP [18], we model our robot as a differential drive with linear and angular velocities $\mathbf{v}_k = [v_k, \omega_k]$, using the kinematic model $\mathbf{X}_{k+1} = f(\mathbf{X}_k, \mathbf{v}_i)$ to propagate the $\mathrm{SE}(2)$ pose $\mathbf{X}_k$ over interval Δt.

The motion model factor enforces the kinematic model and constrains velocities to change according to the acceleration applied over time Δt:

$$\begin{aligned} &\phi^{\text{model}}_{\text{X}}(\mathbf{X}_k,\mathbf{X}_{k+1},\mathbf{v}_k,\mathbf{v}_{k+1}) \propto \\ &\quad \exp\left\{-\frac{1}{2}\left\|\mathbf{X}_{k+1}^{-1} f(\mathbf{X}_k,(\mathbf{v}_k+\mathbf{v}_{k+1})/2)\right\|^2_{\Sigma^{\text{X}}_{\text{model}}}\right\}, \\ &\phi^{\text{model}}_{\text{v}}(\mathbf{v}_k,\mathbf{v}_{k+1},\mathbf{a}_{k+1}) \propto \\ &\quad \exp\left\{-\frac{1}{2}\|\mathbf{v}_{k+1} - (\mathbf{v}_k + \mathbf{a}_k\Delta t)\|^2_{\Sigma^{\text{v}}_{\text{model}}}\right\}. \end{aligned} \tag{6}$$

Constant acceleration factor: This factor encourages robot to smoothly accelerate and decelerate. It is given as:

$$\phi^{\text{const}}(\mathbf{a}_k,\mathbf{a}_{k+1}) \propto \exp\left\{-\frac{1}{2}\|\mathbf{a}_{k+1} - \mathbf{a}_k\|^2_{\Sigma_{\text{const}}}\right\}. \tag{7}$$

Acceleration cost factor: The acceleration cost factor promotes constant velocity trajectory and it is defined as:

$$\phi^{\text{cost}}(\mathbf{a}_k) \propto \exp\left\{-\frac{1}{2}\|\mathbf{a}_k\|^2_{\Sigma_{\text{cost}}}\right\}. \tag{8}$$

Control limit factor: The control limit factor penalizes the planned actions that cross the imposed limits: The hinge loss cost function is defined as:

$$\mathbf{h}_{\text{lim}}(\mathbf{u}_k) = \begin{cases} \mathbf{u}^j_{\text{min}} - \mathbf{u}^j_k & \text{if } \mathbf{u}^j_k < \mathbf{u}^j_{\text{min}} \\ \mathbf{u}^j_{\text{max}} - \mathbf{u}^j_k & \text{if } \mathbf{u}^j_k > \mathbf{u}^j_{\text{max}} \\ 0, & \text{otherwise} \end{cases}, \tag{9}$$

where $\mathbf{u}_k = \{\mathbf{v}_k, \mathbf{a}_k\}$, $\mathbf{u}^j$ denotes the linear or angular component, and $\mathbf{u}^j_{\text{min}}$ and $\mathbf{u}^j_{\text{max}}$ are its lower and upper limits. Accordingly, the control limit factor is:

$$\phi^{\text{lim}}(\mathbf{u}_k) \propto \exp\left\{-\frac{1}{2}\|\mathbf{h}_{\text{lim}}(\mathbf{u}_k)\|^2_{\Sigma_{\text{lim}}}\right\}. \tag{10}$$

Goal factor: The goal factor imposes cost on the final planned pose $\mathbf{X}_N$, ensuring the planned trajectory makes robot progress towards its desired location and orientation. It leads the robot towards the local goal defined ahead of the robot on the global plan. The goal factor is given as:

$$\phi^{\text{goal}}(\mathbf{X}_N) \propto \exp\left\{-\frac{1}{2}\|\mathbf{X}_N - \mathbf{X}_{\text{goal}}\|^2_{\Sigma_{\text{goal}}}\right\}. \tag{11}$$

Dynamic obstacle factor: The dynamic obstacle factor penalizes any robot position close to another object. The d_{ros} represents the preferred safety distance between two

evaluated positions:

$$\epsilon = d_{\text{ros}} - \text{range}(\mathbf{X}_k, \mathbf{L}_k),$$
$$\mathbf{h}_{\text{d-obs}}(\mathbf{X}_k, \mathbf{L}_k) = \begin{cases} \epsilon, & \epsilon > 0 \\ 0, & \text{otherwise} \end{cases}, \tag{12}$$
$$\phi^{\text{d-obs}}(\mathbf{X}_k, \mathbf{H}_k) \propto \exp\left\{ -\frac{1}{2} \|\mathbf{h}_{\text{d-obs}}(\mathbf{X}_k, \mathbf{L}_k)\|^2_{\Sigma_{\text{d-obs}}} \right\}.$$

Directed factor: The directed factor introduced by Kliniewski *et al.* [18] allows directional influence on chosen factors within a factor graph. For instance the *dynamic obstacle factor* between $\mathbf{X}_k$ and $\mathbf{H}_k$ affects the $\mathbf{X}_k$ whilst not influencing the $\mathbf{H}_k$. By carefully formulating the factor graph such that "Prediction" and "Planning" modules are solely connected by *directed factors*, as in Fig. 2, the weight tuning of the modules remains independent of each other.

C-Dynamic obstacle factor: The C-Dynamic obstacle factor is the dynamic obstacle factor that penalizes the object prediction that is close to the ego robot plan. It follows the Eq. (12) and assumes that dynamic object will, to some defined degree, try to avoid the ego robot. Additionally, we include the *C-Dynamic obstacle factors* between every pair of predicted objects, modeling the fact that two dynamic objects will avoid collision with one another.

IV. Nav2 MPPI with Dynamic Obstacle Critic

To extend the widely used Nav2 MPPI controller to dynamic environments, we embed DynORecon's [4] per-object ESDF submaps into a time-parametrized MPPI critic plugin, enabling predictive, geometry-aware avoidance of dynamic obstacles. A system overview is provided in Fig. 3. DynORecon maintains an independent small-scale ESDF submap for each tracked dynamic object, encoding distances to object surfaces, alongside a static background map. It incrementally constructs each object's model in its local frame using point cloud observations, camera trajectories and per-object $\mathrm{SE}(3)$ motions.

A. Predictive Pose Propagation

We propagate each object's pose ${}^{W}\mathbf{L}_{obj}$ using the body-frame twist $\boldsymbol{\xi}^b$, derived from the estimated motion, under a constant-velocity assumption. For a time offset Δt:

$${}^{W}\mathbf{L}_{obj}(t_0 + \Delta t) = {}^{W}\mathbf{L}_{obj}(t_0) \cdot \exp_{\mathrm{SE}(3)}(\boldsymbol{\xi}^b \cdot \Delta t) \tag{13}$$

This allows each rollout state to be evaluated against the obstacle's predicted pose when the robot is expected to reach that state, enabling spatio-temporal collision avoidance.

B. Composite ESDF Query

The per-object submaps are combined with the static map into a composite distance field. All candidate points are assembled in the world frame and, for each submap, filtered against its axis-aligned bounding box (computed around the propagated object pose) before being transformed into the object's local frame for the ESDF lookup. The composite distance at any query point is the maximum signed distance across all submaps, corresponding to proximity to the nearest obstacle surface. The maximum aggregation is deliberate: fusing submaps into a global map using a probabilistic weighted sum, as used within each submap [4], would significantly favor the static background map due to the overwhelming number of observations accumulated. Taking the maximum instead is conservative and safe: the nearest obstacle is always acknowledged, regardless of observation count.

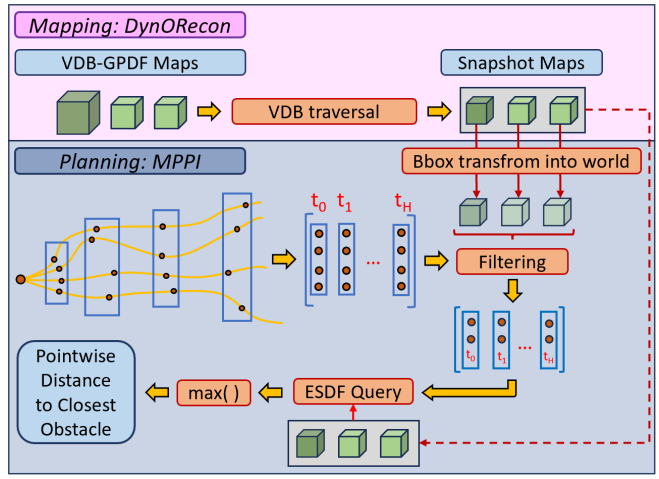

Fig. 3: System overview of the geometry-aware MPPI critic. Mapping and planning are decoupled via a snapshot mechanism. Sampled trajectory states are assembled into batches, filtered against per-object bounding boxes at their predicted future positions, and queried against the corresponding ESDF submaps to compute costs across all rollouts simultaneously.

C. Critic plugin and Trajectory Scoring

The composite ESDF is embedded as a critic plugin that replaces the default obstacle avoidance costs in the Nav2 MPPI stack. The critic maintains asynchronously updated snapshot copies of the ESDF submaps, decoupling the scoring rate from the reconstruction rate. At each sampled trajectory time step, the robot footprint is queried against the composite distance field as four bounding-box corners at sensor height; the maximum SDF value across the corners represents the worst-case proximity to any obstacle surface. Trajectories whose footprint reaches or crosses an obstacle surface are assigned a prohibitive cost; since MPPI selects controls via a softmax weighting over trajectory costs, such trajectories contribute negligibly to the final output. Those entering a configurable soft margin instead receive an additive proportional penalty, allowing the planner to trade off clearance against other navigation objectives.

V. Experiments

This section demonstrates the functionality of the two navigation methods proposed in this work, evaluated in a simulated warehouse environment in Gazebo. All robots are Husarion RosBot XL platforms, controlled via ROS2 as differential drive robots. For all experiments, the global plan for each robot is computed once using the Nav2 default planner on a precomputed static map, and the local goal is selected as the pose approximately 5 meters ahead along

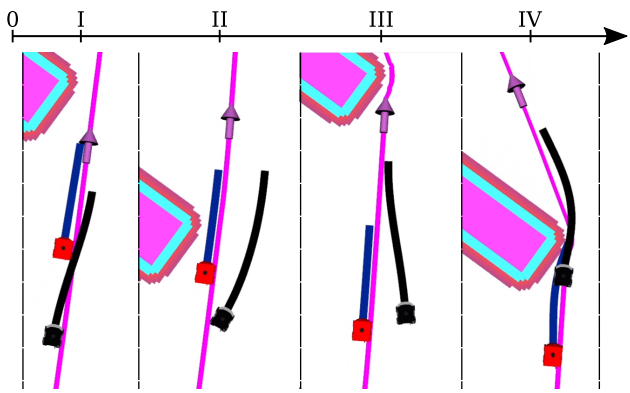

Fig. 4: Experiment illustrating the capability of overtaking the dynamic object (red) by ego robot (black).

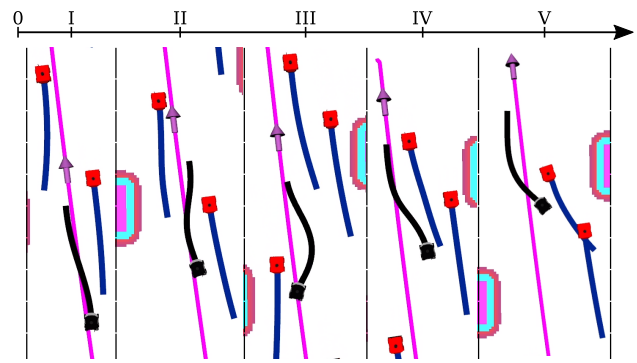

Fig. 5: Experiment illustrating the capability of factor-graph-based MPC controller for navigation in dynamic environments with multiple dynamic objects (red).

the global plan. The table in Fig. 1 describes the color conventions used throughout the experiment figures.

A. Factor-Graph-Based MPC

This section assesses the factor-graph-based MPC described in Section III. It was briefly evaluated by Kliniewski *et al*. [18] where the authors showed its success in navigating dynamic environments. In this section we will further test the controller in different tasks, the decoupled setup and online execution.

The framework is optimized using the Levenberg-Marquardt implementation in GTSAM [29]. The controller uses $\Delta t = 0.1s$ and plans 30 steps ahead ($N = 30$). The optimizer runs at $10\,\mathrm{Hz}$ whereas the commands are executed at $100\,\mathrm{Hz}$. Each time the control command is to be executed, the time elapsed between the current time and optimization start time is calculated and the appropriate command is executed.

In the current setup, we use ground truth robot and object poses perturbed with Gaussian noise to simulate the output of a state estimator. To better inform the controller, real Husarion RosBot XL control command limits and robot size are used. Dynamic objects are controlled independently using the Nav2 MPPI controller with reactive obstacle avoidance based on the GPU lidar.

1) Dynamic Object Overtaking: In this scenario, we spawn the dynamic object ahead of the ego robot, prompt both of them with the same global goal and global planner, and set the velocity limits of the dynamic object to be 20% lower than the ones of ego robot so that the latter naturally needs to overtake. Fig. 4 illustrates the maneuver of overtaking the dynamic object by the ego robot across four time steps.

With the proposed MPC, the robot can reliably overtake the dynamic objects in most situations; however unsuccessful attempts occur when the robot's velocity limits are close to object limits. Additionally, the behavior of the ego robot in Fig. 4 is determined by the chosen goal pose value for the *goal factor* in Eq. (11). Hence a better selection strategy for local goal poses will improve the local planner actions. For example, the local goal selection could consider the dynamic obstacles instead of blindly following the global plan.

2) Multiple Dynamic Object Avoidance: We challenge the robot with a cluttered dynamic environment. The robot in Fig. 5 needs to navigate towards the goal whilst avoiding four objects coming in its direction.

In this configuration, we employed the *C-Dynamic obstacle factor* as explained in Section III, between the ego robot and each of the dynamic objects as well as between every pair of the dynamic objects. Such a setup lets us model the assumption that the ego robot and dynamic objects will all avoid every other dynamic entity in the environment. Importantly, we set the dynamic obstacle safety distance to be two radii of the Husarion Rosbot to enable less conservative driving so that violating safety distance will always lead to collision. The ego robot successfully avoided all the obstacles and navigated towards the goal, which demonstrated that our MPC is suitable for navigation in dynamic environments with multiple dynamic objects.

In this experiment we learned that whilst setting the d_{ros} safety distance between two dynamic obstacles to a small value yields good results, one may often incorporate an additional safety offset. When we increased the d_{ros} by the aforementioned offset, the spherical approximation became highly inaccurate. The robot still achieved the goal; however, its actions were not optimal and could be significantly improved by more accurate object representation.

B. Nav2 MPPI with Dynamic Obstacle Critic

In these experiments, dynamic obstacles are modeled as box-shaped differential drive robots, rather than RosBot XL platforms, to allow greater flexibility in evaluating the effect of obstacle geometry. We conduct two experiments in a narrow corridor of the warehouse environment. In both experiments, the ego robot and a larger dynamic obstacle traverse in opposing directions through this corridor.

We use the MPPI controller as local planner, and it operates at $20\,\mathrm{Hz}$ with $K = 1000$ sampled trajectories and a $5\,\mathrm{s}$ planning horizon. DynORecon provides per-object ESDF submaps with $0.01\,\mathrm{m}$ resolution for dynamic objects and $0.1\,\mathrm{m}$ for the static background. To create a 2D map, we filter out the floor below $z = 0.1\,\mathrm{m}$ and perform ESDF queries at $z = 0.25\,\mathrm{m}$. A soft penalty margin of $0.6\,\mathrm{m}$ is applied around obstacle surfaces, as explained in Section IV-

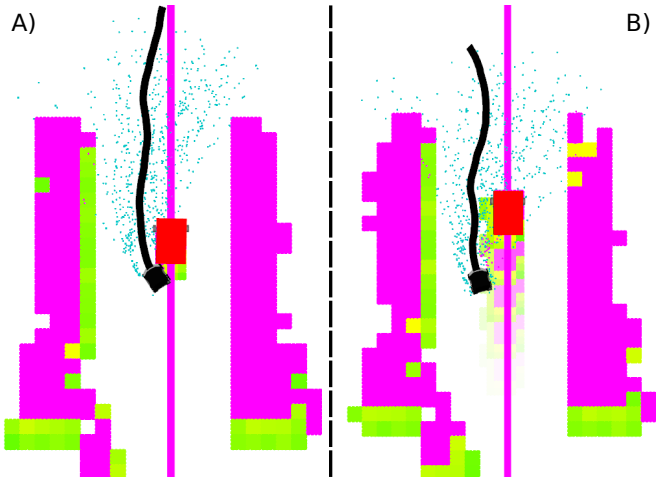

Fig. 6: Effect of temporal projection on dynamic obstacle avoidance. **A)**: Without forward projection, the ego robot (black) reacts too late and collides. **B)**: With constant velocity projection, the ego robot anticipates the obstacle and steers away early.

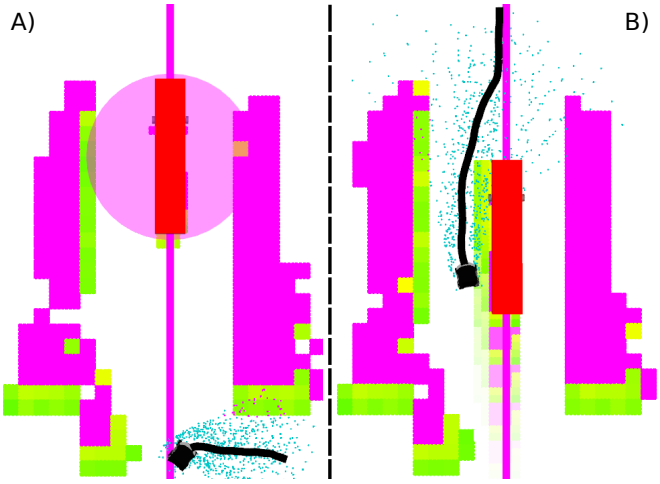

Fig. 7: Effect of geometry-aware obstacle representation in a narrow passage. **A)**: A bounding disc approximation occupies the full corridor width, forcing the ego robot to retreat. **B)**: The geometry-aware ESDF submap preserves lateral clearance, allowing the planner to find a collision-free path.

C. We use the static map purely for global planning, and local obstacle avoidance is realized exclusively by our custom critics based on DynORecon. Evaluations are qualitative, as the scenarios are designed to produce clearly distinguishable outcomes between compared configurations.

1) Effect of Temporal Projection: This experiment evaluates the contribution of temporal projection to anticipatory collision avoidance. The dynamic obstacle is a box-shaped robot with footprint $0.6\times0.4\,\mathrm{m}^2$, navigating towards the ego robot in the narrow corridor as shown in Fig. 6. We compare two avoidance configurations in this experiment: first, the critic evaluates collision costs against the obstacle's current ESDF submap only without forward projection; in the second, the object submap is projected forward in time using the constant velocity model for critic evaluation. All other parameters are identical between the two runs.

As illustrated in Fig. 6, the absence of temporal projection leads to a collision: the ego robot reacts only as the obstacle enters its immediate vicinity, leaving insufficient time and space to maneuver. In contrast, the projection-enabled configuration produces anticipatory behavior: the ego robot begins steering away earlier, successfully avoiding the obstacle with comfortable clearance. This validates the integration of the custom critic into the Nav2 MPPI stack as a viable approach to dynamic obstacle avoidance.

2) Effect of Geometry-Aware Obstacle Representation: This experiment compares navigating a dynamic environment using geometry-aware obstacle representation with primitive shape approximations. The dynamic obstacle is an elongated box-shaped robot of footprint $2.0\times0.4\,\mathrm{m}^2$. To fully encompass the object, the conventional circular approximation requires a large disc as shown in Fig. 7 A). In the comparison experiment, we use the ESDF submap of the object within the critic as proposed. Temporal projection is enabled in both configurations.

As illustrated in Fig. 7 A), using the circular approximation fails to yield a feasible path: the bounding disc radius is large enough to occupy the full corridor width, causing all sampled trajectories to incur prohibitive collision costs and forcing the ego robot to retreat. In contrast, the geometry-aware configuration shown in Fig. 7 B) correctly represents the shape of the obstacle, preserving lateral clearance on either side and allowing the planner to find a collision-free path through. This demonstrates that using incrementally constructed accurate geometry of dynamic obstacles will benefit navigation through challenging tight scenarios where conventional primitive shape approximations can fail.

Together, these experiments validate the proposed critic as a viable Nav2 plugin for dynamic obstacle avoidance. Compared to the factor-graph-based MPC demonstrated in Section V-A, however, the constant velocity prediction model is much simpler and more naive, as it predicts a less smooth or accurate obstacle motion and trajectory.

VI. Conclusion and Future Work

This work presented two novel approaches for navigation in dynamic environments: a factor-graph-based framework that jointly optimizes object trajectory prediction and ego-robot planning, and a geometry-aware MPPI critic that projects per-object ESDF submaps forward in time for predictive collision evaluation within Nav2.

Simulation results demonstrate the effectiveness of both methods. The factor-graph-based approach successfully handles overtaking and multi-obstacle avoidance scenarios, while the MPPI-based approach shows that temporal projection and geometry-aware obstacle representations substantially improve anticipatory collision avoidance and navigation performance.

Future work will focus on incorporating geometry-aware object representations into the factor-graph framework, developing a dynamic-object-aware middle-layer planner, and validating both methods on real Husarion RosBot platforms.

References

[1] B. Bescos, C. Campos, J. D. Tardós, and J. Neira, "Dynaslam ii: Tightly-coupled multi-object tracking and slam," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5191–5198, 2021.

[2] J. Morris, Y. Wang, and V. Ila, "The Importance of Coordinate Frames in Dynamic SLAM," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 13 755–13 761.

[3] J. Morris, Y. Wang, M. Kliniewski, and V. Ila, "DynoSAM: Open-Source Smoothing and Mapping Framework for Dynamic SLAM," *IEEE Transactions on Robotics*, vol. 42, pp. 1178–1197, 2026.

[4] Y. Wang, J. Morris, L. Wu, T. Vidal-Calleja, and V. Ila, "DynORecon: Dynamic Object Reconstruction for Navigation," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 3305–3311.

[5] E. Trevisan, K. A. Mustafa, G. Notten, X. Wang, and J. Alonso-Mora, "Dynamic Risk-Aware MPPI for Mobile Robots in Crowds via Efficient Monte Carlo Approximations," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025, pp. 313–320.

[6] I. S. Mohamed, M. Ali, and L. Liu, "Chance-Constrained Sampling-Based MPC for Collision Avoidance in Uncertain Dynamic Environments," *IEEE Robotics and Automation Letters*, vol. 10, no. 7, pp. 7492–7499, 2025.

[7] M. King-Smith, P. Tsiotras, and F. Dellaert, "Simultaneous Control and Trajectory Estimation for Collision Avoidance of Autonomous Robotic Spacecraft Systems," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 257–264.

[8] Y. Huang, J. Du, Z. Yang, Z. Zhou, L. Zhang, and H. Chen, "A Survey on Trajectory-Prediction Methods for Autonomous Driving," *IEEE Transactions on Intelligent Vehicles*, vol. 7, pp. 652–674, 2022.

[9] B. Lindqvist, S. S. Mansouri, A.-a. Agha-mohammadi, and G. Nikolakopoulos, "Nonlinear MPC for Collision Avoidance and Control of UAVs With Dynamic Obstacles," *IEEE Robotics and Automation Letters*, vol. 5, pp. 6001–6008, 2020.

[10] D.-N. Ta, M. Kobilarov, and F. Dellaert, "A factor graph approach to estimation and model predictive control on Unmanned Aerial Vehicles," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2014, pp. 181–188.

[11] B. Bazzana, T. Guadagnino, and G. Grisetti, "Handling constrained optimization in factor graphs for autonomous navigation," *IEEE Robotics and Automation Letters*, vol. 8, pp. 432–439, 2023.

[12] B. Bazzana, H. Andreasson, and G. Grisetti, "How-to Augmented Lagrangian on Factor Graphs," *IEEE Robotics and Automation Letters*, vol. 9, pp. 2806–2813, 2024.

[13] S. Bari, X. Wang, A. Schoha Haidari, and D. Wollherr, "Factor Graph-Based Planning as Inference for Autonomous Vehicle Racing," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 5, pp. 380–392, 2024.

[14] M. N. Finean, L. Petrović, W. Merkt, I. Marković, and I. Havoutis, "Motion planning in dynamic environments using context-aware human trajectory prediction," *J. of Robotics and Autonomous Systems*, vol. 166, p. 104450, 2023.

[15] M. N. Finean, W. Merkt, and I. Havoutis, "Predicted composite signed-distance fields for real-time motion planning in dynamic environments," in *Proc. of the Conf. on Automated Planning and Scheduling*, vol. 31, 2021, pp. 616–624.

[16] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, "Motion Planning as Probabilistic Inference using Gaussian Processes and Factor Graphs," in *Robotics: Science and Systems XII*. Robotics: Science and Systems Foundation, 2016.

[17] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots, "Continuous-time gaussian process motion planning via probabilistic inference," *Intl. J. of Robotics Research*, vol. 37, no. 11, pp. 1319–1340, 2018.

[18] M. Kliniewski, J. Morris, Y. Wang, I. R. Manchester, and V. Ila, "Dynojepp: Joint estimation, prediction and planning in dynamic environments," *arXiv preprint arXiv:2605.12897*, 2026.

[19] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Information-Theoretic Model Predictive Control: Theory and Applications to Autonomous Driving," *IEEE Transactions on Robotics*, vol. 34, pp. 1603–1622, 2018.

[20] S. Macenski, F. Martin, R. White, and J. Ginés Clavero, "The marathon 2: A navigation system," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[21] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2017, pp. 1366–1373.

[22] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013.

[23] L. Schmid, O. Andersson, A. Sulser, P. Pfreundschuh, and R. Siegwart, "Dynablox: Real-time detection of diverse dynamic objects in complex environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 10, pp. 6259 – 6266, 2023.

[24] L. Schmid, M. Abate, Y. Chang, and L. Carlone, "Khronos: A unified approach for spatio-temporal metric-semantic slam in dynamic environments," in *Proc. of the Robotics: Science and Systems (RSS)*, 2024.

[25] B. Xu, W. Li, D. Tzoumanikas, M. Bloesch, A. Davison, and S. Leutenegger, "Mid-fusion: Octree-based object-level multi-instance dynamic slam," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5231–5237.

[26] J. Mccormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger, "Fusion++: Volumetric object-level slam," in *Proc. of Intl. Conf. on 3D Vision*, 2018, pp. 32–41.

[27] M. Henein, J. Zhang, R. Mahony, and V. Ila, "Dynamic SLAM: The Need for Speed," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2020, pp. 2123–2129.

[28] J. Morris, Y. Wang, and V. Ila, "Online Dynamic SLAM With Incremental Smoothing and Mapping," *IEEE Robotics and Automation Letters*, vol. 11, pp. 3406–3413, 2026.

[29] F. Dellaert and GTSAM Contributors, "borglab/gtsam," May 2022. [Online]. Available: https://github.com/borglab/gtsam