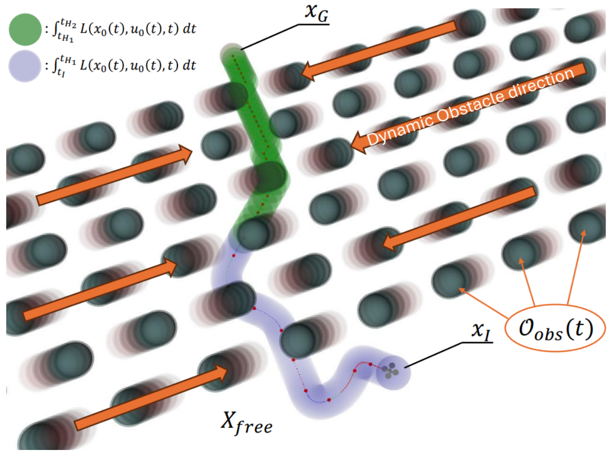
Safe Lattice Planning for Motion Planning with Dynamic Obstacles

Emil Wiman† and Mattias Tiger

***Abstract*— Motion planning in dynamic and uncertain real-world environments remains a critical challenge in robotics, as it is essential for the effective operation of autonomous systems. One strategy for motion planning has been to introduce a state lattice where pre-computed motion primitives can be combined with graph-based search methods to find a physically feasible motion plan. However, introducing lattice planning into dynamic, uncertain settings remains challenging. It is non-trivial to incorporate uncertain dynamic information into the planning process in real time. Thus, in this paper we propose a lattice planning framework for dynamic environments with extensions to handle safety-critical edge-cases that can arise with the uncertain nature of the environment. The proposed method, Safe Lattice Planner (SLP), extends the Receding-Horizon Lattice Planner (RHLP) with enhanced replanning and survival capabilities to handle the dynamic habitat. We thoroughly evaluate SLP in a new benchmark suite against provided baselines. SLP is found to outperform the baselines in terms of safety and resilience in the dynamic environment while reaching the goal state in an efficient manner. We release the benchmark and SLP to accelerate the field of safe robotics.**



● : $\int_{t_{H_1}}^{t_{H_2}} L(x_0(t), u_0(t), t)\, dt$
● : $\int_{t_I}^{t_{H_1}} L(x_0(t), u_0(t), t)\, dt$
x_G
Dynamic Obstacle direction
x_I
$\mathcal{O}_{obs}(t)$
X_{free}

Fig. 1. Challenging motion planning scenario. The agent successfully navigates through a crowd of dynamic obstacles while finding a collision-free, cost-efficient and dynamically-feasible trajectory. Faded spheres denote future dynamic obstacle predictions. Legend is found in Sec. III and IV-B.

I. INTRODUCTION

Safe motion planning is becoming increasingly important when developing autonomous systems meant to inhabit the real-world. Motion planning is the problem of computing a trajectory for a robot from a start state to a goal state while ensuring certain constraints are satisfied throughout the motion [1]. Such constraints can be to avoid collisions and satisfy certain velocity, distance, or time constraints. In this setting, being safe means not posing a threat to other static or dynamic obstacles in the environment, which is achieved by enabling the agent to be aware of its surroundings while constructing its plan. The world we live in is ever-changing. If robots are to share this world with us, they need to navigate safely so that neither property, humans, nor other agents are harmed. By being aware of its surroundings, a robot can incorporate that information into its plan and take appropriate safety measures. Motion planning in an offline or static setting has been extensively studied in the literature [2][3][4] with applications in automotive [5], quadrotors [6] and field-robotics [7]. However, for real-world deployment in complex and unstructured environments, static planning is not feasible due to the temporal and dynamic nature of the real world. Constructing motion plans in dynamic environments is an open problem in robotics as it often requires real-time online planning with limited time budgets for plan construction. The use for safe motion planning is vast, with applications in healthcare [8], at construction sites [9], in agriculture [10], in automotive industry [5], and underwater maintenance [11].

However, existing techniques suffer from not being able to achieve real-time performance in dynamic environments, mainly due to limited time-budgets [12]. Addressing this problem is essential since the real world contains high geometric complexity and a multitude of different dynamic obstacles. Fig. 1 shows an illustrative example of a situation that has been too complex for past approaches, but which is a far cry from what regularly occur in the real world. Autonomous systems that can handle dynamic obstacles are far more useful than those that can not.

In this work, we consider the problem of safe motion planning in complex and unstructured environments in the presence of multiple dynamic obstacles. In this setting we want to navigate as quickly as possible while satisfying system dynamics requirements as well as safety requirements. The contributions of this work are:

- An extensive benchmark[1] suite specifically tailored for evaluating motion planners in dynamic environments.
- An evaluation of existing dynamic motion planners in the provided benchmark. The motion planners are evaluated to provide insight into their different strengths and weaknesses.
- Building on [13], we propose *Safe Lattice Planner* (SLP), a new motion planner that introduces adaptive replanning, local planning, and emergency trajectories

E. Wiman and M. Tiger are with the Department of Computer Science, Linköping University, Sweden. †Corresponding author: emil.wiman@liu.se

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

[1]https://github.com/emilcw/safelatticeplanning

to handle the safety-critical edge-cases in dynamic environments in real-time. SLP demonstrates both preeminent resilience and safety compared to baselines, making strides towards deployable, safe motion planning for the real world.

The paper's organization is as follows: II introduces related research to contextualize the paper's contributions. The dynamic motion planning problem is outlined in III. The presentation of the proposed method, SLP, is in IV. In V an evaluation of the provided baselines compared with the proposed approach is provided. Finally, conclusions in VI.

II. Related Work

The motion planning problem in dynamic environments has been under active research for over four decades [14][15]. The earliest approaches consider motion planning in simple 2D polygon environments with known easily computed obstacle trajectories [14]. Over time, a myriad of different approaches have emerged to tackle the dynamic aspects of the problem, among them modeling obstacles as velocity obstacles [15], introducing probabilistic strategies to handle the uncertain environment [16] and utilizing partial motion planning to achieve real-time performance [17].

Lattice planning in the context of motion planning is derived primarily from the automotive domain [5][3][18]. The lattice framework consists of modeling the problem domain as a lattice where pre-computed feasible motion primitives are used to connect nodes in the lattice. Consequently, the motion planning problem can then be solved online by employing graph-based search algorithms that traverse the lattice with the help of the motion primitives, leading to a fully feasible motion plan. This approach of utilizing lattice planning has been investigated in both static [2] and dynamic [13] environments as well as in structured [19] and unstructured [20] environments. The approach suggested in [20] also proposes the use of a multi-resolution state lattice which is crucial to perform efficient high-quality planning in real-time. This technique has been adapted in [13] which this work builds upon.

The approach in [13] extends the multi-resolution framework by using numerical optimal control [21] to construct locally optimal motion primitives. This ensures that the motion primitives comply with the system dynamics, that physical constraints are met and that smooth control signals can be used all while minimizing a desired performance measure. Generating motion primitives offline for quadrotors have been done before [4][22], however not in a dynamic context or with wait-time states as in [13]. More recent works for trajectory planning have begun to consider dynamic obstacles tracking and prediction [6], however, their evaluations often lack diverse environments to truly test performance.

Previous approaches such as [22] utilize RRT-variants as baselines. The Rapidly-exploring Random Trees (RRT) [23] has been widely used both in a path planning and motion planning context. One such variant of RRT is RRT* [24] which has been shown to be asymptotically optimal by adding a rewiring step to the planning process. Time-based RRT [25] (Temporal RRT) has been used as a baseline in this work but with the RRT* extension. The main extension in Time-based RRT is to extend the configuration space with a time variable to allow for temporal planning.

III. Problem Statement

Consider a robot that is modeled as a time-invariant nonlinear system

$$\dot{x}(t) = f(x(t), u(t)) \tag{1}$$

where $x(t) \in \mathbb{R}^n$ denotes the robot's states and $u(t) \in \mathbb{R}^m$ its control signals. The robot is assumed to have physically imposed constraints on its states $x(t) \in \mathcal{X}$ and control signals $u(t) \in \mathcal{U}$. The 3D-world $\mathcal{X}$ contain volumes that are occupied by obstacles $\mathcal{O}_{\text{obs}}(t)$ and the world changes over time as dynamic obstacles move about. The free-space is where the robot is not in collision with any obstacle at time t and is defined as $\mathcal{X}_{\text{free}}(t) = \mathcal{X} \setminus \mathcal{O}_{\text{obs}}(t)$, see Fig 1. $\mathcal{A}$ is the set of available actions to traverse $\mathcal{X}$.

The solution to the generation of feasible and collision-free reference trajectory from x_I to x_G for (1) is given by solving the dynamic motion planning problem (DMPP) [13],

$$\begin{aligned} \min_{u_0(\cdot),\ t_G} \quad & J = \int_{t_I}^{t_G} L(x_0(t), u_0(t), t)\, dt \\ \text{subj. to} \quad & \dot{x}_0(t) = f(x_0(t), u_0(t)), \\ & x_0(t_I) = x_I, \quad x_0(t_G) = x_G, \\ & x_0(t) \in \mathcal{X}_{\text{free}}(t),\ \forall t \in [t_I, t_G] \\ & u_0(t) \in \mathcal{U},\ \forall t \in [t_I, t_G] \end{aligned} \tag{2}$$

which is a nonlinear nonconvex optimal control problem. The subscript 0 in $x_0(t)$ and $u_0(t)$ distinguishes planned state and control from current state $x(t)$ and control $u(t)$. In addition to (2), the continual planning must remain safe. The objective of the dynamic motion planner is, additionally, to

O1: generate a feasible, collision-free reference trajectory

$$\text{plan} = (x_0(t), u_0(t)),\ t \in [t_I, t_G]$$

that moves the vehicle from its current state x_I to a desired goal state x_G, while optimizing a given performance measure J, e.g., minimum time. L denotes the momentary performance for every time point.

O2: continuously plan to stay collision-free, for Δ_{safe} seconds into the future, at x_G.

O3: re-plan if the current plan becomes infeasible.

O4: if no feasible plan to x_G can be found, plan to get as close as possible to x_G, spatially or temporally.

O5: if no plan can be found that satisfy the robot being collision-free for Δ_{safe} seconds into the future, then plan to avoid collisions for as long as possible.

IV. Proposed Approach

In this work we propose SLP, that extends the work of RHLP [13][26]. To make the paper self-contained, Sec. (IV-A)-(IV-C) summarize the fundamental workings of RHLP. Then, Sec. (IV-D)-(IV-G) introduces the new concepts and improvements that define SLP.

A. Lattice approximation

Finding even a feasible solution to the DMPP, let alone an optimal one, is often intractable. Specialized motion planning algorithms are instead used to achieve real-time performance [1]. To this end, the DMPP (2) is approximated with search over a temporal lattice. The Lattice approximation, first introduced in [26], to the DMPP, is formally

$$\begin{aligned}\min_{a_1,\dots,a_N,\,N} \quad & J = \sum_{n=1}^{N} J_n \qquad (3)\\ \text{subj. to} \quad & \big(x_0^n(t), u_0^n(t)\big) = a_n \in \mathcal{A}, \quad \forall n,\\ & \big|\big|\mathbf{N}x_0^n(0) - \mathbf{N}x_0^{n-1}(t_F^n)\big|\big|_2 = 0, \quad \forall n > 0,\\ & \mathbf{T}(x_I)\; x_0^0(0) = x_I,\\ & \mathbf{T}_{N-1}\; x_0^N(t_F^N) = x_G,\\ & \mathbf{T}_{n-1}\; x_0^n(t) \in X_{\text{free}}(\mathcal{T}_{n-1} + t),\ \forall t \in [0, t_F^n],\ \forall n\end{aligned}$$

where $\mathbf{T}(x)$ is a translation matrix defined by the position in state x and $\mathbf{N}$ is a diagonal matrix with zeros at the position dimensions (projecting a state's position to zero under multiplication). $\mathbf{T}_K = \mathbf{T}(x_I)\prod_{k=0}^{K}\mathbf{T}\big(x_0^k(t_F^k)\big)$ is the resulting translation of the first K motion primitives in the plan and $\mathcal{T}_K = t_I + \sum_{k=0}^{K} t_F^k$ is the start time of the K:th motion primitive in the plan. Here, t_F^k denotes the K:th motion primitive duration. The resulting plan $(t_I, x_I, a_0, \dots, a_N)$ consists of a sequence of N motion primitive actions, $a_0, \dots, a_N$, $\forall a_n \in \mathcal{A}$. The end time of the plan is $t_G = \mathcal{T}_N$. The reference trajectory $(x_0(t), u_0(t)), t \in [t_I, t_G]$ is constructed from the plan by sequential spatio-temporal concatenation of the sequence of motion primitives in the plan. Lattice-based motion planning is resolution complete, and resolution optimal if an admissible heuristic is used with A* [18]. Lattice-based motion planning is consequently equivalent to the DMPP in the resolution-limit.

B. Multiple Receding Horizon Lattice Planning

Traditional receding horizon motion planning (RHMP) divides the temporal horizon of J into two parts: an initial short horizon where the full problem is solved and a cost-to-go term Φ estimating the remaining cost of getting to the goal. A simple estimate like weighted Euclidean distance makes the planner prone to dead-ends, while global planning remains intractable. Instead [13] strike a balance between global and local planning, by dividing the temporal interval of DDMP into three parts,

$$J = \underbrace{\int_{t_I}^{t_{H_1}} L(x_0(t), u_0(t), t)\, dt}_{\text{High-resolution temporal lattice}} + \underbrace{\int_{t_{H_1}}^{t_{H_2}} L(x_0(t), u_0(t), t)\, dt}_{\text{Low-resolution lattice}} + \underbrace{\int_{t_{H_2}}^{t_G} L(x_0(t), u_0(t), t)\, dt}_{\approx \Phi(x_0(t_{H_2}), x_G, t_{H_2})}.$$

The first part is solved in a high-resolution temporal lattice where all motion primitives in $\mathcal{A}$ are allowed, including high-velocity and other high-performing maneuvers. The second part is solved using a low-resolution lattice, where a very limited set of motion primitives are allowed, $\mathcal{A}_{\text{reduced}} \subset \mathcal{A}$, see Fig. 1. The last part is again the estimated cost-to-go denoted Φ. The second and third part form a systematic way to estimate the cost-to-go, which can now be informative enough to let the planner avoid poor long-term trajectory selections. If a solution to the goal is found already in the low-resolution lattice of the second part, then the resulting global plan is a physically feasible trajectory from start to goal. As the robot moves closer to the goal then the transition from high-resolution to low-resolution lattice moves forward in a receding horizon fashion. This plan construction is compliant with Sec. III **O1**.

C. Safe and Proactive Motion Planning

The horizon of the first part is required to be collision-free with dynamic obstacles for at least Δ_{safe} seconds into the future, adding the constraint

$$t_{H_1} - t_I \geq \Delta_{\text{safe}} \qquad (4)$$

to the optimization problem. This part ensures that the motion plan is safe and effective in the short term and that it is consistent with Sec. III **O1-O2**. Since the motion of obstacles far into the future is harder to predict, only collisions with static obstacles are considered in the low-resolution lattice search. If the motion planner has reached its goal x_G it will plan to remain there using wait-time state (as introduced in [13], a wait-time state allows the planner to intentionally idle the robot) while fulfilling the requirement of having a safe plan Δ_{safe} seconds into the future. That is, in a pure waiting scenario we set $x_G = x_I$ and produce a plan of duration Δ_{safe} (so $t_G = t_I + \Delta_{\text{safe}}$) to ensure the robot remains safe while idle. t_I is the time-point of the start of the plan. In a stochastic world, a valid plan can become unsafe, so the robot continuously replans to avoid collisions and reach its goal, a process known as dynamic in-flight replanning [27]. This part is connected to Sec. III **O3**.

D. Temporal Motion Planning

The receding-horizon lattice-based motion planning problem is solved using a two-stage graph search (Alg. 1). The first search is halted if it reaches a time-out (Δ_{phase1} seconds) before finding the goal (Alg. 1, line 9). If so, then the second search begins from all safe partial plans in the frontier of the first search. It does so with an initial transition from $\mathcal{A}$ to $\mathcal{A}_{\text{reduced}}$ and then continuing with only using $\mathcal{A}_{\text{reduced}}$ until the goal is found or a second time-out (Δ_{phase2}) is reached (Alg. 1, line 26). If no plan to the goal is found, the lattice planner operates as a local planner with a very informative cost-to-go estimator. The resulting plan will have one of the statues in Tab. I, which enumerates the possible outcomes statues which are also annotated in Alg. 1 for clarity.

Status	Description
EMERGENCY	Collision before reaching next state on lattice, escape with emergency trajectory to safe state, see Sec. IV-G.
FULL	A global temporal plan is found to the goal.
REDUCED	A plan with a temporal horizon (of at least Δ_{safe}) after which a reduced set of motion primitives are used. A global plan unless accompanied by the LOCAL status.
EXHAUSTED	Search space exhausted, no plan found to the goal.
LOCAL	The goal was not found. Utilize the frontier to find a plan where the end of the plan is the temporally longest collision-free plan ending up closest to the goal or $x_{collision}$, see Sec. IV-F.
EPHEMERAL	A plan with a temporal horizon that is less than Δ_{safe}. Consequently, the plan is not guaranteed to be collision-free up until Δ_{safe}. A global plan unless accompanied by the LOCAL status.
FAILURE	No collision-free plan can be found.

TABLE I

POSSIBLE RESULT CATEGORIES OF SLP, ALG. 1.

Furthermore, a found plan that is either REDUCED or EPHEMERAL can, in addition, be LOCAL if the goal is not found in the second search using the low-resolution lattice. Also, temporal motion planning requires some prediction component in order to predict the future trajectories of dynamic obstacles. In this work, only simple constant velocity [28] prediction has been implemented for the sake of simplicity. However, the planning algorithm is agnostic to this and it can easily be extended with a more sophisticated approach as future work.

E. Adaptive Replanning

In [13], the replanning time (e.g. sum of Δ_{phase1} and Δ_{phase2}) is set to a fixed constant. This is impractical in a dynamic environment if the planner assumes it has, say, 3 seconds to replan but a collision will occur in 2 seconds, then a collision is inevitable unless the replanning interval adjusts adaptively. To address this issue, an adaptive replanning algorithm has been developed in SLP, see Fig. 2. Inspecting Fig. 2, in (1) there is no future collision detected and thus the replanning time is set to $\Delta_{safe} \cdot \epsilon$ where $\epsilon \geq 0$ depends on the platform and the available sensors. In (2), a future collision is detected but $t_{collision} > \Delta_{safe}$, hence the replanning time is set to Δ_{safe}. In (3), a collision will occur within Δ_{safe} and the replanning time is accordingly set to $t_{collision} \cdot \gamma$ where $1 \geq \gamma \geq 0$ is tuned depending on preferred safety measures and the temporal length of motion primitives. Lastly in (4), if the duration of the plan is too short, set the replanning time to the temporal length of the plan. In this work, $\epsilon = 2$ and $\gamma = 2/3$, found by manual tuning until sufficient behavior. This ensures that Sec. III **O3** can be done in a feasible way.

Algorithm 1: Receding-Horizon Temporal SLP function for making a plan from a given state to a goal state.

```
1  function PLAN(plan, $x_G$, $t_{start}$, $\Delta_{safe}$, $\Delta_{phase1}$, $\Delta_{phase2}$, $\mathcal{A}$, $\mathcal{A}_{reduced}$):
2  emergency, $t_{collision}$, $x_{collision}$ $\leftarrow$ IN_COLLISION(plan, $t_{\text{start}}$)
3  if emergency then
4      plan $\leftarrow$ BREAKOUT(plan, $t_{\text{collision}}$)
5      return plan, EMERGENCY
6  end
7  $x_I \leftarrow$ ADAPTIVE(plan, $t_{collision}$, $\Delta_{safe}$)
8  explored $\leftarrow \{\}$, frontier $\leftarrow \{x_I\}$, $\mathbb{X}_{\text{goal}} \leftarrow \{x_{\text{G}}\}$
9  plan $\leftarrow$ SEARCH($t_{\text{start}}$, $X_{\text{goal}}$, $\Delta_{\text{phase1}}$, $\Delta_{\text{safe}}$, $\mathcal{A}$, frontier, explored)
10 if $|plan| > 0$ then
11     $(t', x') \leftarrow plan[\text{end}]$
12     if $x_G = x'$ and $t' \geq t_{start} + \Delta_{\text{safe}}$ then
13         return plan, FULL
14     end
15     if frontier $= \{\}$ or $t' < t_{start} + \Delta_{safe}$ then
16         if frontier $= \{\}$ then
17             frontier $\leftarrow \{(t, x) \in$ explored $\mid t = \max_t$ explored$\}$
18         else if $t' < t_{start} + \Delta_{safe}$ then
19             frontier $\leftarrow \{(t, x) \in$ frontier $\mid t = \max_t$ frontier$\}$
20         end
21         status $\leftarrow$ EPHEMERAL
22     else
23         frontier $\leftarrow \{(t, x) \in$ frontier $\mid t \geq t_{\text{start}} + \Delta_{\text{safe}}\}$
24         status $\leftarrow$ REDUCED
25     end
26     plan $\leftarrow$ SEARCH($t_{\text{start}}$, $X_{\text{goal}}$, $\Delta_{\text{phase2}}$, $\mathcal{A}_{\text{reduced}}$, frontier, explored)
27     $(t', x') \leftarrow plan[\text{end}]$
28     if $x_G = x'$ then
29         return plan, status
30     else if frontier $= \{\}$ then
31         return plan, EXHAUSTED
32     else
33         $x_{\text{local}} \leftarrow$ LOCAL-PLANNING(frontier,
34         $x_{collision}$, $x_G$, $\Delta_{safe}$)
35         plan $\leftarrow$ GENERATE-PLAN($x_{\text{local}}$, frontier, explored)
36         return plan, (status, LOCAL)
37     end
38 else
39     return $\{\}$, FAILURE
40 end
```

F. Local planning

If no collision-free plan is available but a collision isn't imminent, the robot shouldn't stay idle. Instead, it should engage in best-effort emergency collision avoidance, as described in LOCAL (Tab. I), hoping the environment changes to allow a new safe plan. Alg. 1, line 33, enables a survival strategy when planning fails, whether the robot is trapped or planning is interrupted (line 9 or 26). From the previous search, several sub-plans exist that can be used for best-effort behavior. These sub-plans either fall under the statues REDUCED or EPHEMERAL. In this work, the best-effort

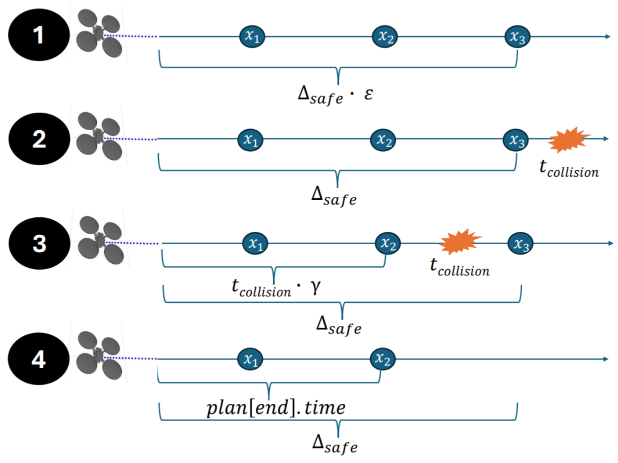


Fig. 2. Decision-making process in the adaptive replanning component. Item 1-4 implies different resulting replanning times. x_i denotes a state on the lattice.

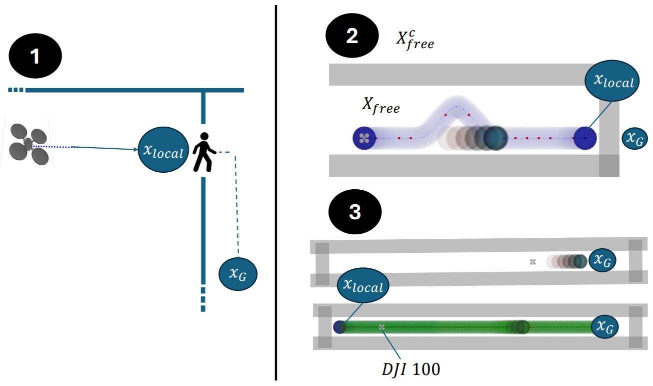


Fig. 3. Local planning (1) A plan has existed previously but is no longer feasible, plan to stay as close to the collision state as possible to await an opening (2) No plan exists to the goal due to static obstacles, plan to get as close as possible while avoid dynamic obstacles (3) Dynamic collision is unavoidable, the agent tries to find the temporally longest collision-free plan.

emergency collision avoidance strategy consists of two steps. First, if there exists a previous plan (meaning that this is not the first planning cycle) where a dynamic collision has occurred and the time of collision is $t_{\text{collision}} > \Delta_{\text{safe}}$, then the plan that takes us the closest to the collision state (without actually colliding of course) and keeps us alive the longest is selected:

$$x_{\text{local}} = \min_{||x - x_{\text{collision}}||_2} \texttt{frontier} \tag{5}$$

The reasoning behind this strategy is that if there has previously existed a plan that leads to the goal and this plan becomes invalidated at some state, then the planner should try to maintain its presence close to that state since it might become available in the near future given the dynamic environment, see Fig. 3, item 1. If no dynamic collision has occurred beforehand, the best-effort emergency collision strategy reduces to:

$$x_{\text{local}} = \min_{||x - x_{\text{G}}||_2} \texttt{frontier} \tag{6}$$

The presented strategy prioritizes survival time and being as close as possible to the goal, see Fig. 3 item 2. If the robot is fully boxed in, it will stay away from colliding with any obstacles as long possible as in Fig. 3 item 3 (see Sec. IV-C). Depending on application, other strategies might be more suitable. This extension addresses Sec. III **O4**-**O5**.

G. *Emergency trajectories*

A limitation concerning RHLP is that if a dynamic collision were to occur during a primitive execution, it cannot be avoided since replanning only can be done from states on the lattice. In a dynamic setting this can be highly dangerous since collision might occur during primitive execution. In this work we propose the use of emergency trajectories, see Fig. 4, to address this issue. If at any point in the primitive execution $t_{\text{collision}} < primitive\_end\_time$ is fulfilled (Fig. 4 (1)), then the planner will find a suitable breakout point further in time on the current primitive but before $t_{collision}$. From this point, an expansion of a limited set of primitives will be made and the one the keeps the robot collision-free for the longest duration will be selected. This emergency primitive will be connected to the breakout point and the robot will escape the imminent collision (Fig. 4 (2)). As a consequence, the robot will end up in an off-lattice pose. In the next planning iteration the robot will find the closest on-lattice pose, steer to this state using the NMPC and resume the replanning process (Fig. 4 (3)). This feature primarily addresses Sec. III **O5** but with even narrower constraints on safe planning time.

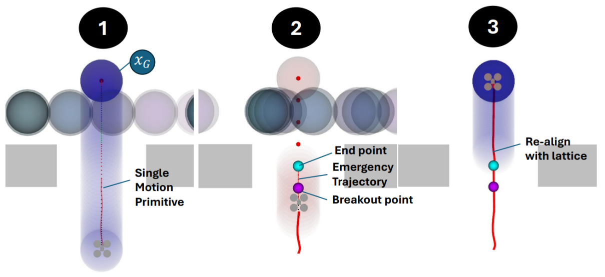


Fig. 4. Emergency trajectory. (1) During primitive execution is a collision detected before reaching the next state. (2) Abort primitive execution from breakout point and escape to end point. (3) Re-align with lattice and restart the planning process. In this scenario two dynamic obstacles are blocking the opening. Note that faded spheres are estimated future positions.

V. EXPERIMENTAL RESULTS

To examine the performance of SLP compared to other dynamic motion planners, particularly in challenging complex scenarios, has a benchmark been provided (V-A). The benchmark consists of four different motion planners, RRT* [24], Temporal RRT* (T-RRT*) [29], RHLP [13] and the proposed approach SLP. The planners undergo initial assessment in static environments to collect baseline performance before they are put in a dynamic setting. Note that RRT* is considered a static motion planner (does not possess temporal planning capabilities) while the rest are dynamic in the sense that they can consider dynamic information in the planning process.

#	Scenario	No. Dyn. Obs.	Static/Dynamic	Obs. Type
1	Empty	0	Static	None
2	Wall	0	Static	None
3	Wall2	0	Static	None
4	Wall3	0	Static	None
5	Dead End	0	Static	None
6	Blind Corner	2	Both	RHLP
7	Crosswalk	2	Both	CV
8	Crosswalk2	60	Both	CV
9	Crosswalk3	120	Both	CV
10	Guard	1	Both	RHLP
11	Corridor	1	Both	CV
12	Corridor2	2	Both	RHLP
13	Warehouse Empty	0	Static	None
14	Warehouse1	10	Both	RHLP
15	Warehouse Large	11	Both	RHLP
16	Warehouse2	10	Both	RHLP
17	Warehouse3	10	Both	RHLP
18	Unreachable	1	Both	CV
19	Survival	1	Both	CV

TABLE II
SCENARIOS PART OF THE BENCHMARK.

A. Benchmark

The proposed benchmark[1] includes nineteen different scenarios (see Tab. II) where some scenarios are static and some are dynamic. All of the scenarios vary in size, geometrical appearance and number of dynamic obstacles, see Fig. 5. For dynamic obstacle movement, either a constant velocity motion model [28] is used or a simplified version of RHLP (using only $A_{reduced}$ no wait-time and no predictions). The current position and velocity of the dynamic obstacles are known at all times to the motion planner from the simulation. The benchmark code is containerized utilizing Docker [30] and the code is implemented in C++ utilizing ROS [31]. This simplies running the benchmark on different machines despite different requirements on ROS and other dependencies. The robotic platform used in all scenarios is the DJI Matrice 100[2] and it utilizes the same nonlinear modell as in [13]. The nonlinear MPC controller first developed in [32] and later used in [13] has been employed for trajectory tracking and remains the same in all provided motion planners. The motion primitives are generated offline utilizing ACADO [21]. For both RRT* and Temporal RRT*, each node in the resulting branch is sequentially fed to the NMPC, which then steers the system toward a hovering state at each node. In contrast, lattice-based motion planners use motion primitives, providing the NMPC with a complete reference trajectory instead. For the lattice-based motion planners, the lattice discretization is fixed at 0.5 m in all experiments. The experiment procedure is the following:

[2]www.dji.com/matrice100

- For each scenario run: Select a motion planner to be evaluated.
- The agent starts in a scenario-specific location.
- The scenario handler starts the specified scenario and spawns in static and dynamic obstacles. The scenario handler sends a scenario-specific navigation goal to the motion planner.
- The motion planning algorithm starts and planning begins towards the supplied navigation goal.
- The experiment will continue until the agent reaches the desired navigation goal or until a hard time limit of 900 seconds is reached.

Each scenario is repeated ten times for each planner, and the results are reported as $\mu \pm \sigma$ in each scenario. The results discussed in Sec. V-B and V-E are aggregated over several scenarios, where the results in V-C and V-D are not. The aggregated results in Table III show combined mean and standard deviation across diverse scenarios with varying size, geometry, and dynamic obstacles. High variability causes the standard deviation to sometimes exceed the mean. Despite this, it can still indicate a planner's overall performance. The randomness in each scenario comes from non-deterministic computations. The supplied performance measures are **TCD**: Total Collision Duration [s], **NOC**: Number of (dynamic) collisions, **TUC**: Time until (first) collision [s], **PT**: Planning Time [s], **PL**: Path Length [m] and **SR**: Success Rate [%] reached goal out of the possible navigation goals.

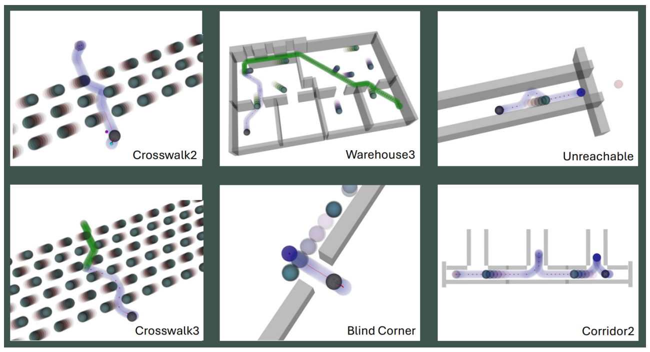


Fig. 5. Selected scenarios from the provided benchmark used in the static and dynamic evaluation in Sec. V-B and V-E.

B. Static scenarios

Initial experiments were conducted in a static setting to assess the motion planners in a non-dynamic setting. The aggregated results can be found in Tab. III *Static Environment*. We selected a subset of the most challenging scenarios for dynamic evaluation in Sec. V-E; accordingly, we evaluate those same scenarios in static mode here (scenarios 6, 8, 9, 12, 17, 18; see Fig. 5). From Tab. III *Static Environment* it can be observed that RRT* and T-RRT* suffer from increased **PT** and **PL**, compared to the lattice-based motion planners, while not achieving full **SR**. RHLP and SLP perform similarly and in all metrics, which is expected due to the static setting of the environment.

Motion Planner	TCD			NOC			TUC			PT			PL			SR
	Mean	±	STD	Mean	±	STD	Mean	±	STD	Mean	±	STD	Mean	±	STD	
Static Environment (Scenario 6, 8, 9, 12, 17 and 18 Tab. II)																
RRT*	0	±	0	0	±	0	inf	±	0	1.92	±	3.98	44.4	±	36.56	47/60
T-RRT*	0	±	0	0	±	0	inf	±	0	1.65	±	3.89	45.82	±	36.92	48/60
RHLP	0	±	0	0	±	0	inf	±	0	**0.67**	±	**1.25**	**29.06**	±	**30.44**	**50/60**
SLP	0	±	0	0	±	0	inf	±	0	0.97	±	2.32	31.45	±	34.03	**50/60**
Unreachable (Scenario 18 Tab. II)																
RRT*	9.99	±	0.04	1	±	0	27.55	±	0.03	10.01	±	0.11	0	±	0	0/10*
T-RRT*	9.99	±	0.01	1	±	0	27.54	±	0.01	9.79	±	0.10	0	±	0	0/10*
RHLP	10	±	0.02	1	±	0	27.53	±	0.01	**3.02**	±	**0.01**	0	±	0	0/10*
SLP	**0**	±	**0**	**0**	±	**0**	**inf**	±	**0**	5.77	±	0	**19.87**	±	**0.93**	0/10*
Survival (Scenario 19 Tab. II)																
RHLP	9.49	±	1	1	±	0	**84.47**	±	**0.62**	0.02	±	0.001	67.1	±	0.3	10/10
SLP	**7.93**	±	**0.30**	1	±	0	82.84	±	0.09	**0.01**	±	**0.001**	**61.78**	±	**0.4**	10/10
Blind Corner (Scenario 6 Tab. II)																
RRT*	1.22	±	0.16	1	±	0	6.02	±	0.13	0.03	±	0	12.73	±	0.02	10/10
T-RRT*	1.56	±	0.41	1.8	±	0.4	5.99	±	0.16	0.01	±	0	13.73	±	1.54	10/10
RHLP	0.98	±	0.97	1.3	±	0.6	11.52	±	3.95	0.10	±	0.11	**6.50**	±	**2.82**	10/10
SLP	**0.42**	±	**0.85**	**0.2**	±	**0.4**	**14.55**	±	**0.17**	**0.001**	±	**0**	9.81	±	3.44	10/10
Dynamic Environments (Scenario 6, 8, 9, 12, 17 and 18 Tab. II)																
RRT*	26.14	±	31.4	4.87	±	4.68	20.31	±	16.32	1.78	±	4.04	42.15	±	31.24	46/60
T-RRT*	75.35	±	92.21	6.47	±	5.47	85.36	±	101.97	2.64	±	3.7	48.76	±	39.29	34/60
RHLP	4.12	±	5.88	0.92	±	0.95	45.49	±	42.05	**0.92**	±	**1.2**	**31.25**	±	**29.84**	**50/60**
SLP	**1.99**	±	**2.68**	**0.27**	±	**0.23**	**324.56**	±	**446.34**	1.05	±	2.32	42.48	±	31.9	**50/60**

TABLE III

MERGED BENCHMARKING RESULTS FOR DIFFERENT MOTION PLANNERS ACROSS MULTIPLE SCENARIOS. THE TABLE PRESENTS MEAN AND STANDARD DEVIATION FOR THE PERFORMANCE METRICS DESCRIBED IN SEC. V-A. THE RESULTS ARE DISCUSSED IN SEC. (V-B)-(V-E). FOR **SR** IN *Unreachable* (MARKED WITH *) IS IT NOT POSSIBLE TO REACH THE GOAL SINCE IT IS BLOCKED BY STATIC OBSTACLES, THUS IS 0 THE BEST POSSIBLE RESULT. AQUIRING *inf* IN **TUC** DENOTS THAT NO COLLISIONS OCCURRED. **BOLD** MARKS THE BEST POSSIBLE RESULT WHEN ITS NOT A TIE.

C. *Evaluation of Local planning*

To examine the performance of the newly added capabilities in SLP were experiments conducted that specifically targeted these functionalities. First, the introduction of Local planning, see Sec. IV-F, was assessed in the *Unreachable* scenario (see Tab. II No. 18). In the *Unreachable* scenario, no collision-free trajectory exists to the goal (Fig. 3 (2)), while a dynamic obstacle approaches the robot's initial position. This intentionally benchmarks agent survival when no plan to the destination exists. The results can be found in Tab. III *Unreachable*. The baselines (RRT*, T-RRT*, RHLP) cannot handle this scenario as they lack dynamic obstacle avoidance without a feasible goal trajectory, causing collisions. Also no baseline planner moves from the initial position, leading to 0 in **PL**. In contrast, SLP utilizes its local planning capabilities and successfully avoids all collisions while moving as close as possible to the goal. Here, the result in **PL** for SLP is noted as the best, since we favor taking action instead of staying idle in this scenario. Additionally, RHLP and SLP were evaluated in the *Survival* scenario (See Tab. II No. 19). In this scenario, avoiding a collision is impossible (See Figure, 3). Both RHLP and SLP try to maximize **TUC**, see Tab. III *Survival*. Both planners show survival and resilience capabilities by actively avoiding collisions as long as possible.

D. *Evaluation of Emergency trajectories*

Second, the introduction of emergency trajectories, see Sec. IV-G was assessed in the *Blind Corner* scenario (see Tab. II No. 6). This scenario consists of an opening in a door where the goal state is patrolled by two dynamic obstacles moving back and forth, see Fig. 4. This scenario is designed so that the lattice-based motion planners should find a single primitive that will take them to the goal (when the gap is momentarily clear). However, following this primitive will inevitably lead to a collision, thus the primitive execution must be aborted in order to maintain safety constraints. The baselines all fail as shown in Tab. III *Blind Corner* while SLP outperforms them in terms of **TCD** and **NOC**.

E. *Dynamic Scenarios*

Finally, to thoroughly assess the planners, they have been evaluated across multiple dynamic scenarios see Tab. III *Dynamic Scenarios*. Adaptive replanning is primarily evaluated in this setup. This table aggregates the mean (10 runs) for each scenario in Fig. 5 and showcases the "average" performance for each motion planner in this benchmark suite.

Inspecting Tab. III it can be noticed that SLP achieves the lowest **TCD** and **NOC** compared to the baselines while maintaining a high **TUC**. This suggests that it effectively avoids collisions overall but also tries to postpone them until no other alternative exists, demonstrating resilience. Naturally, this comes with a trade-off in **PT** and **PL** (compare *Static Environment*) but this does not impact the overall performance of reaching the goal, see **SR**.

VI. CONCLUSION

In this work we propose SLP, a safe lattice planning framework built to allow for motion planning in dynamic environments. We extend the work on RHLP by formalizing the Receding-Horizon Temporal planning algorithm and extending it with edge-case functionality such as adaptive replanning, local planning and emergency trajectories. To validate the performance of SLP, we conduct extensive experiments in a set of diverse dynamic scenarios where we also compare against baselines. Our experiments show that SLP outperforms the baselines both in terms of safety and resilience in dynamic scenarios. In future work, we aim to combine SLP with DAEP [29] to perform real-world safe 3D-exploration in dynamic environments.

REFERENCES

[1] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006.

[2] J. Butzke, K. Sapkota, K. Prasad, B. MacAllister, and M. Likhachev, "State lattice with controllers: Augmenting lattice-based path planning with controller-based motion primitives," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 258–265.

[3] C. Katrakazas, M. Quddus, W.-H. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transportation Research Part C: Emerging Technologies*, vol. 60, pp. 416–442, 2015.

[4] M. Pivtoraiko, D. Mellinger, and V. Kumar, "Incremental micro-uav motion replanning for exploring unknown environments," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 2452–2458.

[5] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[6] J. Zhong, M. Li, Y. Chen, Z. Wei, F. Yang, and H. Shen, "A safer vision-based autonomous planning system for quadrotor uavs with dynamic obstacle trajectory prediction and its application with llms," in *2024 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*, 2024, pp. 920–929.

[7] C. Zhou, B. Huang, and P. Fränti, "A review of motion planning algorithms for intelligent robots," *Journal of Intelligent Manufacturing*, vol. 33, pp. 387–424, 2022. [Online]. Available: https://doi.org/10.1007/s10845-021-01867-z

[8] R. Alterovitz and K. Goldberg, *Motion Planning in Medicine: Optimization and Simulation Algorithms for Image-Guided Procedures*, 1st ed. Springer Publishing Company, Incorporated, 2008.

[9] C. Zhang and A. Hammad, "Improving lifting motion planning and re-planning of cranes with consideration for safety and efficiency," *Advanced Engineering Informatics*, vol. 26, no. 2, pp. 396–410, 2012, knowledge based engineering to support complex product design. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474034612000043

[10] G. Colucci, A. Botta, L. Tagliavini, P. Cavallone, L. Baglieri, and G. Quaglia, "Kinematic modeling and motion planning of the mobile manipulator agri.q for precision agriculture," *Machines*, vol. 10, no. 5, 2022. [Online]. Available: https://www.mdpi.com/2075-1702/10/5/321

[11] J. D. Hernández, M. Moll, E. Vidal, M. Carreras, and L. E. Kavraki, "Planning feasible and safe paths online for autonomous underwater vehicles in unknown environments," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 1313–1320.

[12] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2016.

[13] O. Andersson, O. Ljungqvist, M. Tiger, D. Axehill, and F. Heintz, "Receding-horizon lattice-based motion planning with dynamic obstacle avoidance," in *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 4467–4474.

[14] J. Reif and M. Sharir, "Motion planning in the presence of moving obstacles," in *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*, 1985, pp. 144–154.

[15] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[16] M. A. Erdmann, "On probabilistic strategies for robot tasks," 1989. [Online]. Available: https://api.semanticscholar.org/CorpusID:13410578

[17] S. Petti, "Safe navigation within dynamic environments: a partial motion planning approach," 2007. [Online]. Available: https://api.semanticscholar.org/CorpusID:221148082

[18] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *Journal of Field Robotics*, vol. 26, no. 3, pp. 308 – 333, March 2009.

[19] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," in *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, ser. IROS'09. IEEE Press, 2009, p. 1879–1884.

[20] J. Petereit, T. Emter, and C. W. Frey, "Mobile robot motion planning in multi-resolution lattices with hybrid dimensionality," *IFAC Proceedings Volumes*, vol. 46, no. 10, pp. 158–163, jun 2013.

[21] B. Houska, H. J. Ferreau, and M. Diehl, "Acado toolkit—an open-source framework for automatic control and dynamic optimization," *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.

[22] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 2872–2879.

[23] S. M. LaValle, "Rapidly-exploring random trees : a new tool for path planning," *The annual research report*, 1998.

[24] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotic Research - IJRR*, vol. 30, pp. 846–894, 06 2011.

[25] A. Sintov and A. Shapiro, "Time-based rrt algorithm for rendezvous planning of two dynamic systems," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 6745–6750.

[26] M. Tiger, D. Bergström, A. Norrstig, and F. Heintz, "Enhancing lattice-based motion planning with introspective learning and reasoning," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4385–4392, 2021.

[27] M. Wzorek, J. Kvarnström, and P. Doherty, "Choosing replanning strategies for unmanned aircraft systems," in *International Conference on Automated Planning and Scheduling (ICAPS)*, vol. 2, no. 3, 2010.

[28] X. Rong Li and V. Jilkov, "Survey of maneuvering target tracking. part i. dynamic models," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1333–1364, 2003.

[29] E. Wiman, L. Widén, M. Tiger, and F. Heintz, "Autonomous 3d exploration in large-scale environments with dynamic obstacles," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 2389–2395.

[30] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux journal*, vol. 2014, no. 239, p. 2, 2014.

[31] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," vol. 3, 01 2009.

[32] M. Kamel, T. Stastny, K. Alexis, and R. Siegwart, *Model Predictive Control for Trajectory Tracking of Unmanned Aerial Vehicles Using Robot Operating System*. Cham: Springer International Publishing, 2017, pp. 3–39.