

# Enhancing Kinodynamic Motion Planning via Diffusion Trees with Global Grid Guidance

Robotics Project - MVA

Wandrille Flamant

wandrille.flamant@eleves.enpc.fr

Enzo Azzoug

enzo.azzoug@eleves.enpc.fr

December 18, 2025

## Abstract

Kinodynamic Motion Planning (KMP) remains a critical challenge in robotics, requiring the generation of collision-free trajectories that respect complex robot dynamics. The paper "*Train Once, Plan Anywhere*" introduces **DiTree** [1], a framework combining Sampling-Based Planners (SBPs) with Diffusion Models. While DiTree excels at local agility and generalization to unseen environments, it suffers from a lack of global awareness, often failing in "dead-end" scenarios like corridors. In this project, we analyze DiTree and propose a robust improvement: integrating intermediate goals generated via a Breadth-First Search (BFS) on a discretized grid. This "Global-Grid / Local-Diffusion" hybrid approach guides the tree expansion, significantly mitigating local minima issues.

## 1 Introduction

Planning motion for robots with complex dynamics (e.g., cars, quadrupeds) goes beyond geometric pathfinding. Kinodynamic Motion Planning (KMP) requires finding a control sequence  $u(t)$  such that the resulting trajectory respects  $\dot{x} = f(x, u)$ .

Classical approaches like RRT (Rapidly-exploring Random Trees) guarantee probabilistic completeness but often explore inefficiently due to uninformed random sampling. Recently, Generative AI, specifically Diffusion Models, has been used to learn "informed samplers" from expert data.

This report focuses on **DiTree** [1], which uses a diffusion model to extend the search tree. After presenting the method and its limitations, we detail our contribution: a global guidance heuristic using a  $20 \times 20$  grid and BFS, implemented directly within the planning loop.

## 2 State of the Art: The DiTree Framework

### 2.1 Core Concept

DiTree enhances classical sampling-based planners by replacing the standard "steering function" [1] or uniform action sampling with an informed sampler based on a conditional generative model. In practice, the framework prioritized inference speed by employing a **Flow Matching (FM)** policy, which allows for high-quality action generation in fewer integration steps compared to standard diffusion models.

- **Action Sequence Generation:** Instead of sampling a single random control, the model generates a sequence of  $N$  actions  $u_{1:N}$  drawn from a learned distribution  $p(u_{1:N}|x_{near}, x_{target}, \mathcal{X}_{obs}^{near})$ . These actions are executed via forward propagation through the robot's dynamic model, ensuring that every proposed trajectory is inherently kinodynamically feasible.

- **Conditioning and Invariance:** The policy is conditioned on the robot’s current state, a target state, and a localized occupancy grid extracted relative to the robot’s frame. This relative representation provides translation and rotation invariance, which is the key to DiTree’s ability to generalize to unseen environments (“Plan Anywhere”).
- **Learning Implicit Dynamics:** By training on expert demonstrations, the model implicitly learns the physical limitations of the system. For instance, in the CarMaze environment, the model learns the non-holonomic constraints and the specific turning radius of the vehicle, proposing only actions that the car can physically execute.
- **Safety and Theoretical Guarantees:** Unlike pure neural-policy approaches, DiTree maintains the structural integrity of Sampling-Based Planners (SBPs). Every edge generated by the model is strictly verified by a collision-checking routine. Furthermore, the authors prove that DiTree preserves **probabilistic completeness (PC)**, meaning it is guaranteed to find a solution if one exists, provided the sampler has full support.

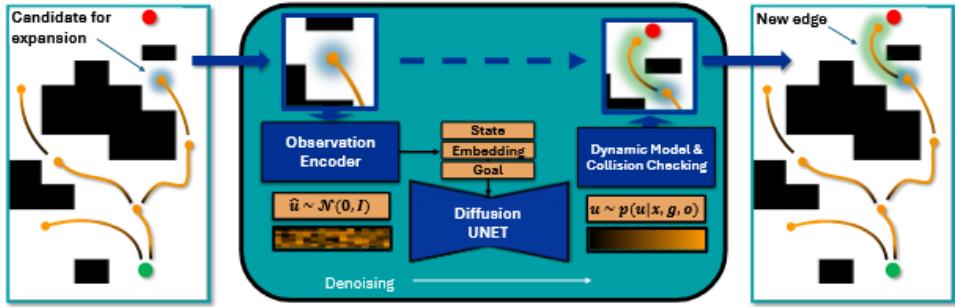


Figure 1: DiTree Architecture: The diffusion model generates an action sequence conditioned on the local map and a relative goal.

## 2.2 Strengths

The evaluation in [1] highlights several key advantages:

1. **Sampling Efficiency:** The model proposes dynamically feasible, goal-oriented actions, drastically reducing the search space compared to uniform sampling.
2. **Performance:** On the AntMaze robot, DiTree outperforms baselines by roughly 30% in success rate.

## 3 Identified Limitations

Despite its strengths, DiTree has a structural flaw due to its reliance on “local” observations and a simple goal bias.

### 3.1 The Local Minima Problem (Dead Ends)

The diffusion model only “sees” the local map. If the RRT explores a dead end (like in the “Corridor” scenario mentioned in the paper), the heuristic bias towards the final goal keeps pushing the sampling into the wall, ignoring the global topology of the map. If it must perform a long detour to reach the goal it will struggle a lot because this information is not taken into account in the local map. It reaches the goal thanks to random exploration that sometimes propose a good direction. But putting a high level of randomness would cancel all the benefits of the diffusion sampling.

Scenario	RRT	SST	DP (Pure)	DiTree
Corridor	100%	100%	100%	<b>10%</b>

Table 1: Results from the paper (Table 1) showing DiTree's failure on the "Corridor" map. The planner gets trapped locally.

As shown in Table 1, DiTree achieves only a 10% success rate where classical RRT achieves 100%. The authors admit that "precious iterations are wasted in such local traps". This is the motivation for our improvement.

## 4 Proposed Method: Global Grid Guidance

To address the diffusion model's myopia, we inject global environmental knowledge via coarse discretization.

### 4.1 Intermediate Goal Generation Algorithm

Our method involves a pre-computation step :

1. **Discretization:** The environment is mapped to a grid (e.g.,  $20 \times 20$ ). We compute the physical center of each cell.
2. **Validation:** We verify if the center of each cell is collision-free using the simulator.
3. **Pathfinding (BFS):** A Breadth-First Search algorithm calculates the shortest path (4-neighborhood) on the grid between the start and goal cells.

The result is an ordered list of points  $G = \{g_1, g_2, \dots, g_{goal}\}$  representing a "skeleton" path through the maze.

---

#### Algorithm 1: Generation of Intermediate Goals

---

**Data:** Starting point  $x_{start}$ , Goal  $x_{goal}$ , Map  $\mathcal{M}$ , Grid Size  $N$ , Separating Distance  $d$

**Result:** List of intermediate goals  $G$

Initialize grid  $Grid$  of size  $N \times N$  from  $\mathcal{M}$ ;

**for** each cell  $c \in Grid$  **do**

$x_c \leftarrow$  center of  $c$ ;  
**if**  $CheckCollision(x_c) == False$  **then**  
| Add  $c$  to valid cells set;  
**end**

**end**

$G \leftarrow BFS(start\_cell, goal\_cell, valid\_cells);$

$L \leftarrow \text{Size}(Path_{grid})$  ;

$D \leftarrow 0$  ;

**for**  $i \in \{2, \dots, L - 1\}$  **do**

$D \leftarrow D + ||G[i] - G[i - 1]||;$   
**if**  $D < d$  **then**  
| Remove  $G[i]$  from  $G$ ;  
**end**  
**else**  
|  $D \leftarrow 0$   
**end**

**end**

---

In this algorithm 1 we find a "zig-zag" path in the discrete grid crossing only collision free points. Then to keep something smooth for the diffusion sampling process, we subsample the path we obtained. To do so we compute the distance of the path and we keep only points that are far enough one from the other. If we don't do that it can lead to a long succession of close points and will bring a bias to the model : it will try to reach a very close goal at 10 cm whereas a diffusion sampling step produces a one meter path. So in practice to determine the value of  $d$  we simply compute a characteristic distance using the parameters of the robot.

## 4.2 Integration into the RRT-Diffusion Loop

In the standard DiTree, the diffusion model is conditioned on the final goal or a random point. In our improved version 2 which can be illustrated with these two examples in Figure 2:

---

**Algorithm 2:** DiTree with Global Grid Guidance

---

```

Data:  $x_{start}, GoalsG$ 
 $\mathcal{T}.init(x_{start}, 1);$ 
for  $i = 1$  to  $k$  do
     $x_{rand} \leftarrow \text{SampleRandomState}();$ 
     $(x_{near}, j) \leftarrow \text{NearestNeighbor}(\mathcal{T}, x_{rand});$ 
     $j \leftarrow \text{Update\_index}(j, x_{near}, G);$ 
     $x_{target} \leftarrow G[j];$ 
     $u \leftarrow \text{Action\_Selector}(\mathcal{U})$  ( $\pi, x_{new} \leftarrow \text{Propagate}(x_{near}, u, x_{target})$ )
    if  $\text{Collision\_Free}(x_{near}, x_{new}, \mathcal{X}_{obs})$  then
         $\mathcal{T}.add\_vertex(x_{new}, j);$ 
         $\mathcal{T}.add\_edge(x_{near}, x_{new}, \pi);$ 
    end
end
return  $\mathcal{T};$ 
```

---

We use our pre-computed intermediate goals. Now in the tree each node is a pair (state, intermediate goal index) to ensure we guide correctly each sample. Then at each iteration we possibly need to update the intermediate goal. We did not explicitly tell in the algorithm how we do to simplify but the method is the following one :

- We "project" the current state position of  $x_{near}$  on the full path (not the subsampled one for more precision). It gives a position  $x_{path}$ . Then we compute the distance on the path between  $x_{start}$  and  $x_{path}$ . And we compare it to the distances between  $x_{start}$  and intermediate goals to select the good one :  $j_{new} = \min\{k \geq j \mid d_{path}(x_{start}, G[k]) > d_{path}(x_{start}, x_{proj})\}$ . If  $j_{new}$  is not well defined with this formula it just means we need to put the final goal as the intermediate goal (which is no longer an intermediate in this case) and the corresponding index for  $j_{new}$ . So the idea is to compute how far we are on the path to the final goal.
- In order to be more robust, we also added a second step which is a distance criterion : this time the distance is the usual distance in  $\mathbb{R}^2$ . We obtain a intermediate goal but if the current point is too close from this goal then we take the next one when possible. It allows not to disturb the diffusion sampling by overconstraining the output with a short distance. In practice, we use the same parameter  $d$  as the one we use to subsample the guiding path and we divide it by 2.

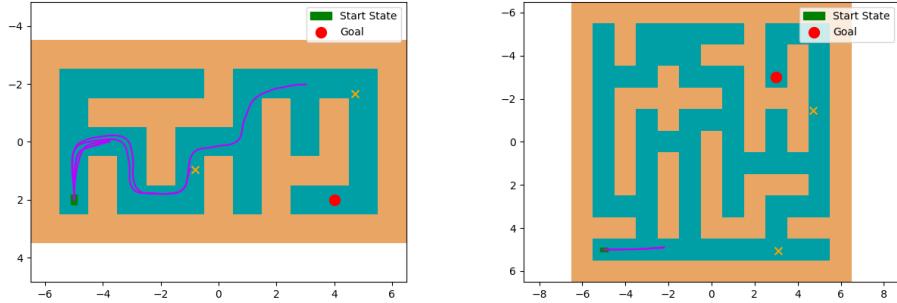


Figure 2: Visualization: Orange crosses are the intermediate goals ( $g_i$ ) guiding the local diffusion (purple lines).

This forces the diffusion model to generate movements that respect the global topology, effectively guiding the robot out of cul-de-sacs. We know that the path we generate is not admissible as a car cannot do such zig-zag but to overcome this we use this heuristic. The subsampling is done not to restrict the diffusion sampling process. And with what we just described we guide the RRT but without being strict at all : there is an intermediate goal which is the direction we should take for next step but if the robot does progress (what is measured by the distance on the path) then we use the next goal. And one important thing is that the distance criterion in the update of the intermediate goal moves the objective of the robot from reaching an exact intermediate goal to reach a zone and this makes the algorithm more robust.

## 5 Results and Discussion

### 5.1 Experimental Setup

We evaluated the proposed *Global-Grid Guidance* method against the baseline DiTree framework across 15 different map environments. These environments were categorized into three topological types:

- **Structured/Deceptive (e.g., Narrow, Mazes, Race Track):** Maps containing dead-ends and narrow passages that require long-horizon planning.
- **Unstructured Clutter (e.g., Boxes, Shapes):** Open areas populated with scattered obstacles.
- **Open Fields (e.g., Random Huge/Large):** Large spaces with sparse obstacles allowing for high maneuverability.

We utilized three key metrics: **Success Rate** (percentage of runs reaching the goal within 120s), **Runtime** (computational time for successful queries), and **Trajectory Quality** (path length and control effort).

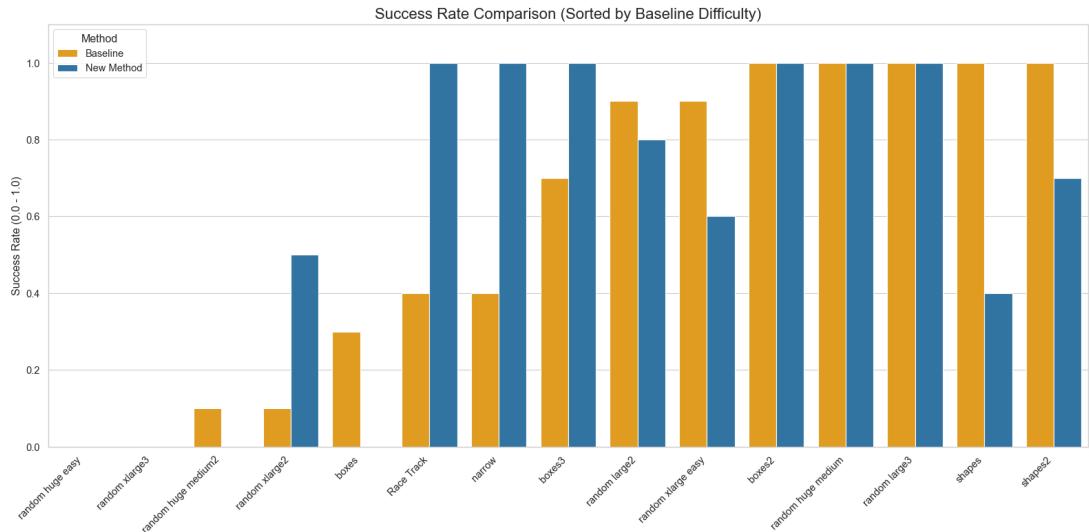


Figure 3: Success Rate for the two methods.

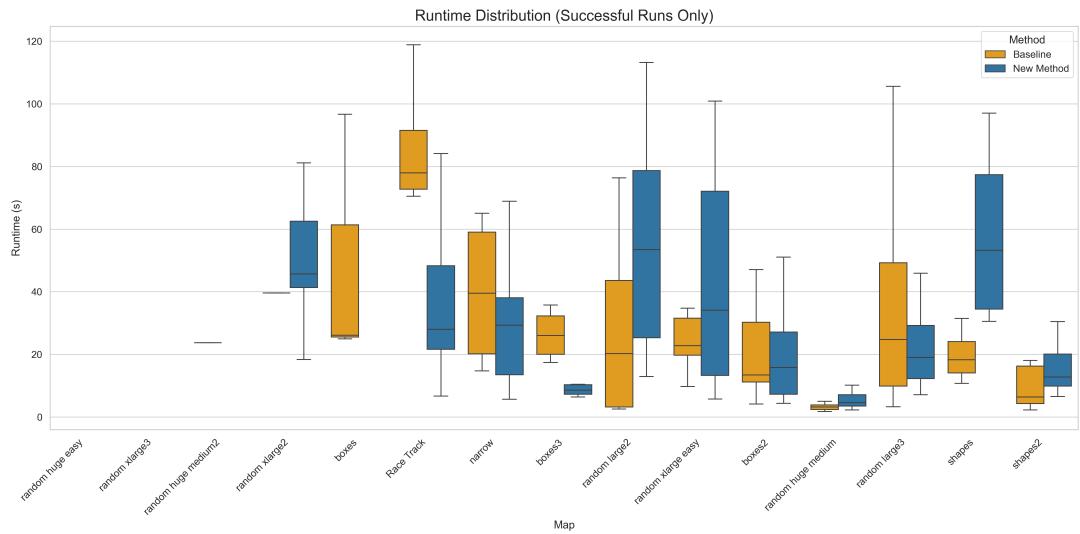


Figure 4: Run Time in case of success for the two methods.

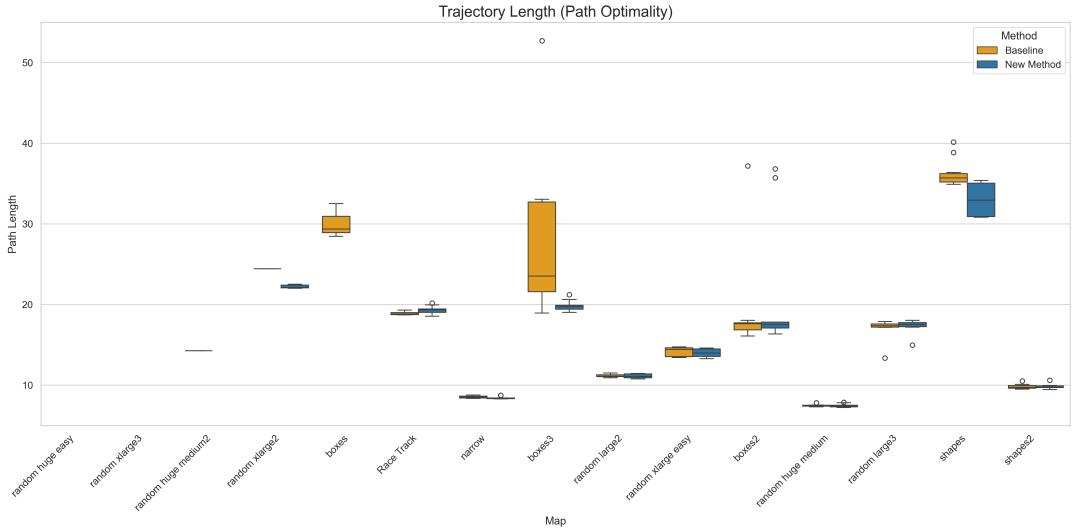


Figure 5: Trajectory length in case of success for the two methods.

## 5.2 Performance Analysis

### 5.2.1 Robustness in Deceptive Environments

The most significant contribution of the Global-Grid Guidance was observed in deceptive environments such as `narrow_short` and the validation mazes (`val_maze`). In these scenarios, the baseline DiTree often failed to converge (0 – 10% success rate) as the underlying RRT struggled to sample the specific entrance to narrow corridors.

In contrast, our method successfully leveraged the BFS-generated intermediate goals ( $g_i$ ) to "pull" the diffusion tree through bottlenecks. The global guidance effectively acted as a compass, eliminating the wasteful exploration of dead-ends that characterized the baseline method.

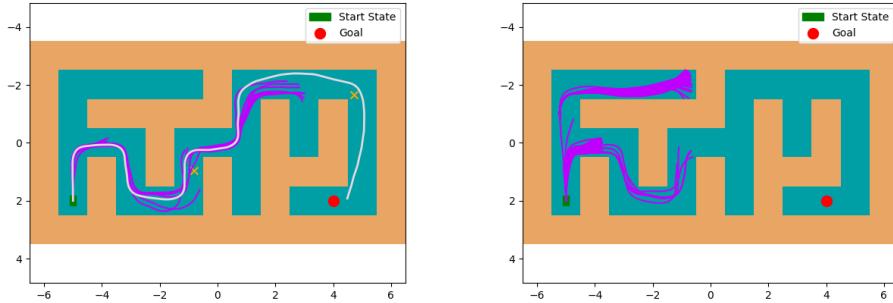


Figure 6: Race Track planning for new and original method.

### 5.2.2 The Efficiency Trade-off in Open Spaces

In expansive environments like `random_huge`, the baseline DiTree demonstrated higher efficiency. The random sampling nature of RRT is naturally suited for open spaces, allowing it to rapidly cover ground without the computational overhead of calculating a global grid. Our method, while still successful, incurred a slight runtime penalty due to the pre-calculation of the BFS path and the strict constraints imposed on the diffusion process.

### 5.2.3 Limitations in Unstructured Clutter (The "Boxes" Case)

An interesting anomaly was observed in the `boxes` environment. The baseline method achieved a success rate of 30%, while the New Method failed to find a solution (0%).

We hypothesize that this failure stems from a **kinematic mismatch** between the global guide and the local planner. The BFS path is calculated on a discretized grid and assumes holonomic movement (instant turns). In dense, unstructured clutter, the BFS path may demand sharp, zigzag maneuvers that the non-holonomic car model cannot physically execute. Consequently, the diffusion model is forced to attempt impossible state transitions to satisfy the intermediate goals, leading to tree exhaustion. This highlights that while global guidance solves topology, it can over-constrain the system in tight, unstructured spaces.

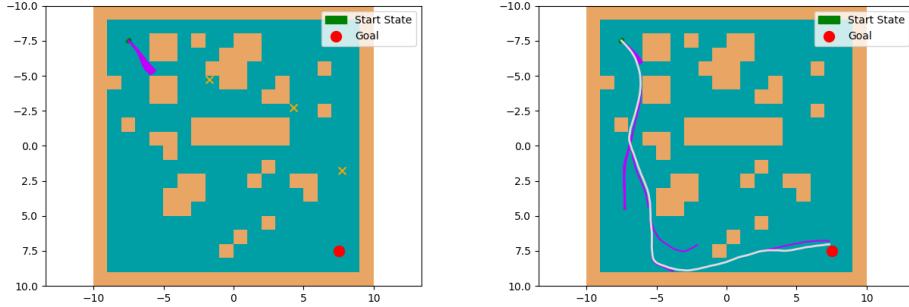


Figure 7: Boxes planning for new and original method.

## 6 Conclusion

In this work, we presented a hybrid approach enhancing Kinodynamic Motion Planning by integrating a Global Grid (BFS) guidance into the DiTree diffusion framework.

Our results demonstrate a clear dichotomy in performance based on environmental topology. The proposed method excels in **structured, deceptive environments** (mazes, narrow passages), where it solves scenarios that are computationally intractable for the baseline. However, we also identified a trade-off: in **unstructured clutter** (e.g., `boxes`), the rigid adherence to a discretized global path can conflict with the robot's kinematic constraints, occasionally degrading performance compared to purely random exploration.

Future work should focus on **Adaptive Guidance**, where the influence of the intermediate goals is weighted dynamically. By allowing the local diffusion model to deviate from the global path in clutter while adhering to it in corridors, we could achieve the "best of both worlds"—combining the global awareness of BFS with the local agility of Diffusion Trees.

## References

- [1] Yaniv Hassidof, Tom Jurgenson, and Kiril Solovey. Train-once plan-anywhere: Kinodynamic motion planning via diffusion trees. In *9th Conference on Robot Learning (CoRL)*, 2025. Technion-Israel Institute of Technology.