Team Spirit: Dynamic Obstacle Avoidance for Quadrupeds

Tariq Anwaar, Aditya Bharambe, Sahil Chaudhary, Richa Mohta, Kaustabh Paul

***Abstract*—Quadruped robots are increasingly employed in scenarios demanding high mobility and adaptability, such as disaster response, exploration, and industrial inspection. While existing frameworks like Quad-SDK provide a robust foundation for locomotion and control, they often fall short in handling dynamic obstacles and complex terrains that require real-time adaptability. This paper extends Quad-SDK by integrating advanced Z-axis planning and dynamic obstacle avoidance using a Probabilistic Roadmap (PRM) coupled with A*-based search algorithms. The proposed system ensures kinodynamically feasible paths that account for velocity constraints, terrain traversability, and obstacle geometry. Experimental validation demonstrates that the enhanced framework enables quadrupeds to navigate dynamic, unstructured environments effectively, showcasing improved adaptability and robustness in real-world scenarios.**

***Index Terms*—Dynamic Obstacle Avoidance, Collision Checking, PRM, weighted A*, Global Planning**

I. Introduction

Quadruped robots have become a focal point of research due to their ability to navigate unstructured and complex terrains. Their unique locomotion capabilities allow them to operate effectively in environments where wheeled or tracked robots struggle, making them invaluable for search-and-rescue missions, industrial inspections, and exploration tasks. As robotic systems are increasingly deployed in real-world scenarios, the demand for reliable, adaptive locomotion has grown significantly.

Dynamic environments present unique challenges, requiring quadrupeds to avoid moving obstacles, adapt to uneven terrain, and replan paths in real-time. Existing systems often rely on pre-defined trajectories and frame-based planning methods, which are susceptible to delays, noise, and limited responsiveness during aggressive or unexpected maneuvers. Overcoming these challenges is critical to advancing quadruped capabilities for reliable operation in diverse settings.

This work addresses these challenges by extending Quad-SDK, a widely used open-source framework for quadruped locomotion. By integrating dynamic obstacle avoidance and Z-axis planning, this project enhances the framework's ability to navigate complex terrains. The proposed enhancements aim to enable quadrupeds to adapt dynamically to their surroundings, ensuring safe and efficient navigation in real-world environments.

II. Related Work

Quad-SDK, introduced by Norby et al. in 2022, provides a comprehensive software framework for quadruped robotics, encompassing state estimation, trajectory generation, and control mechanisms [6]. This modular architecture has significantly contributed to advancing quadruped locomotion, particularly in structured and semi-structured environments. However, the global planning component of Quad-SDK, relying on RRT-Connect, exhibits limitations in dynamic environments where real-time adaptability is paramount.

Norby and Johnson (2020) further explored motion planning for dynamic legged robots by proposing a hierarchical approach combining global planning with reactive control [5]. While this method effectively reduced computation overhead and enhanced path efficiency, its reliance on static planning frameworks constrains its performance in scenarios requiring continuous environmental updates and rapid obstacle avoidance.

Fig. 1: Spirit Quadruped

Probabilistic Roadmaps (PRM) have emerged as a robust solution for global motion planning, particularly in high-dimensional spaces. PRM constructs a graph of feasible paths by randomly sampling configurations and connecting them based on collision-free edges. PRM faces challenges due to the need for frequent updates. Lazy PRM, introduced by Bohlin and Kavraki, mitigates this by deferring collision checks until a path is queried, significantly reducing computational load in environments with sparse obstacles [1]. These advancements in PRM and its derivatives inspire our approach to dynamic obstacle avoidance and Z-axis planning, where adaptability and efficiency are critical.

Dynamic obstacle avoidance in robotics has seen significant strides with real-time methods like D* Lite and its variants. Koenig and Likhachev proposed D* Lite as an efficient re-planning algorithm for dynamic environments, making it suitable for scenarios involving frequently changing terrains [3]. Furthermore, work by Ferguson et al. introduced Anytime D*, which enables bounded suboptimal solutions for planning in real-time applications [2]. These algorithms provide critical

insight into the adaptive planning necessary for quadruped navigation.

Collision checking has been a central focus in robotic planning, with bounding box-based methods providing computationally efficient means of detecting and avoiding obstacles. Sliding window techniques, such as those discussed by Van Den Berg et al., leverage temporal and spatial information to enhance obstacle detection accuracy during navigation [4]. By integrating these methodologies, this work seeks to ensure robust and reliable operation in complex terrains.

III. Background

A. Quad-SDK

Quad-SDK, developed by the Robomechanics Lab, is an open-source framework designed to support research in quadruped robotics. It offers a modular architecture that facilitates trajectory generation, state estimation, and control. Researchers and practitioners leverage this framework to experiment with advanced locomotion and manipulation algorithms in both simulated and real world environments. Its integration with common robotic middleware enables a streamlined development and testing of quadruped systems.

Despite its robustness, the Quad-SDK faces limitations in handling dynamic and complex environments. Its current implementation lacks support for real-time Z-axis planning and dynamic obstacle avoidance, restricting its utility in scenarios involving significant environmental variations. Additionally, pre-defined trajectories often fail to account for sudden changes in terrain or obstacle movement, limiting the framework's adaptability.

This project builds on Quad-SDK's foundation, addressing these limitations by incorporating advanced planning algorithms and real-time adaptability. By focusing on dynamic obstacle avoidance and kinodynamically feasible Z-axis planning, the work extends Quad-SDK's capabilities to meet the demands of real-world, unstructured environments, thus advancing the state of quadruped robotics. The system architecture is shown in fig 2.

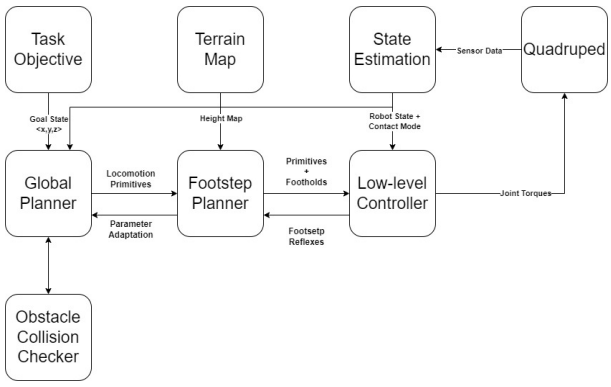


Fig. 2: System Architecture

B. Collision Checking

Collision checking in motion planning ensures that planned paths are safe from potential collisions with static and dynamic obstacles. The current QUAD-SDK framework has a built-in terrain map which is 2.5D in nature (i.e, for every combination of x and y, the map will have a particular z values). Hence, this 2.5D terrain map representation cannot be used to visualize obstacles which have x, y, and z coordinates. In this work, the authors rely on bounding box approximations to model the geometry of both the quadruped robot and the surrounding obstacles. This approach strikes a balance between computational efficiency and sufficient accuracy for real-time planning.

To handle moving obstacles, a sliding window approach has been implemented, that incorporates dynamic updates within the planning pipeline. This method uses a temporal horizon to predict the future states of moving obstacles based on their velocity and trajectory. The robot's bounding box is updated in real-time within this sliding window, enabling the system to anticipate potential collisions and re-plan as necessary.

While the bounding box approach is computationally lightweight, its effectiveness depends on careful tuning of the bounding box dimensions to avoid overly conservative or permissive planning. The sliding window mechanism addresses the challenge of dynamic environments by providing predictive collision checks to ensure the system is responsive to changes in obstacle positions. By combining bounding boxes with a sliding window approach, the collision-checking module enhances the robot's ability to navigate safely and efficiently in dynamic, unstructured environments.

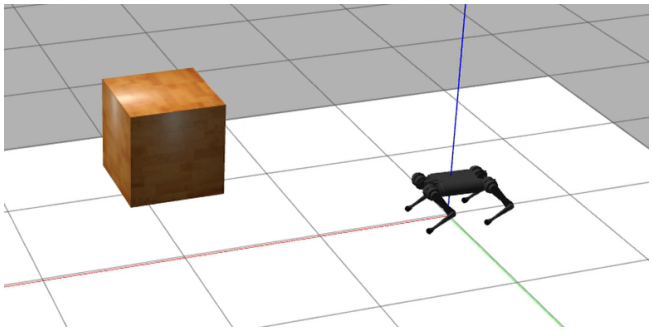


Fig. 3: The simulation environment

IV. Planning Algorithm

A. Lazy PRM

A PRM (Probabilistic Roadmap) is a widely used sampling-based planning algorithm that generates a graph-based representation of a robot's configuration space (known as the map). It works by generating a fixed number of random nodes within the environment and connecting these nodes based on collision-free paths. This roadmap serves as a representation of feasible motion paths through the environment. With a generated map and a given start and goal state, a search-based planner like A* can be used in conjunction to retrieve a path, if it exists. This two-step process leverages a precomputed map for path finding. However, the PRM generation process is computationally expensive and hence performed offline. The advantage lies in the fact that once generated, the graph can be used to find paths to any combination of start and goal states. Typically, while generating the PRM, sampled nodes that lie in

obstacles are rejected, ensuring that the graph represents only collision-free paths. However, this approach is only effective in static environments, as any changes in the environment will necessitate regeneration of the PRM. Fig. 1 represents the planning environment with the block representing the static obstacle. To represent a dynamic obstacle, a uniform velocity is applied to the block.

To address this limiation, a modification to PRM for dynamic environments is called Lazy PRM. Unlike traditional PRM, this approach does not reject nodes that lie within obstacles. Instead, it retains them in the graph, but with an infinite cost. This modification ensures that the roadmap remains valid even if the environment changes. When an obstacle moves or disappears, the cost of the corresponding nodes can be dynamically updated rather than requiring a complete regeneration of the graph. This small modification enables PRMs to be used in dynamic environments. The cost updation and re-planning is handled by the A*, or for a faster and more optimal re-planning algorithm, D* Lite. These capabilities make Lazy PRM a powerful extension of the PRM framework, enabling its use in dynamic and uncertain environments where adaptability is crucial.

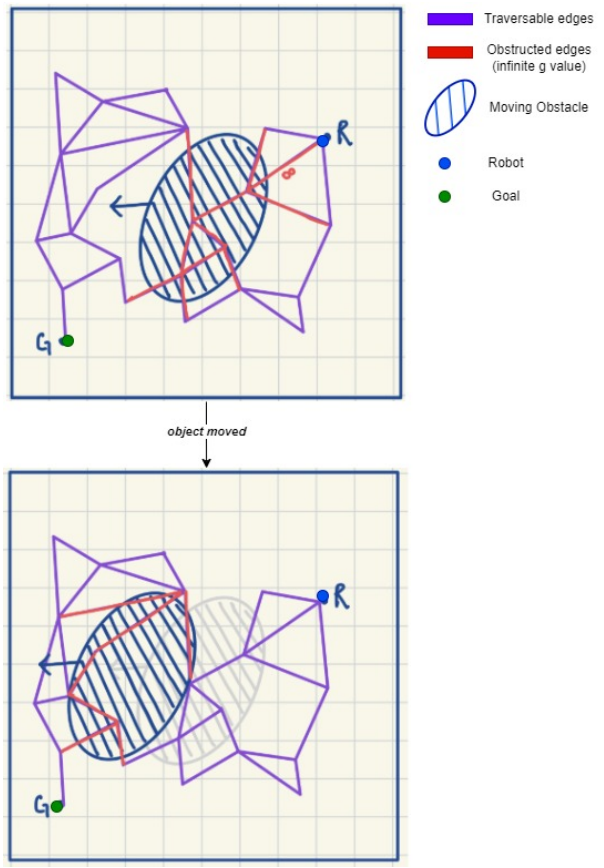


Fig. 4: Edge costs getting updated as obstacle is moving through the Lazy PRM roadmap

In this paper, Lazy PRM is implemented to retain nodes within dynamically changing obstacles, enabling efficient updates to the roadmap by dynamically adjusting edge costs during re-planning as shown in fig. 4. There is a modular code structure which utilizes key data structures such as adjacency lists for graph representation, priority queues for A*-based path finding, and unordered sets for managing open and closed node lists.

Run	Number of Samples	Vertices	Build Time (s)
1	5000	3435	1.013
2	10000	5550	2.925
3	20000	8573	7.735

TABLE I: PRM Roadmap Generation

B. A* and Weighted A*

Given a graph, the A* search guarantees an optimal path from a start state to a goal state, if it exits. This theoretical guarantee of A* is what makes it so effective and hence an ideal candidate for searching through the PRM. The g value and heuristic used is shown in equation 1 and 2 respectively.

$$g(s') = g(s) + \|\mathbf{s}_2.\text{pos} - \mathbf{s}_1.\text{pos}\|_2 \cdot w_{\text{pos}} + \|\mathbf{s}_2.\text{vel} - \mathbf{s}_1.\text{vel}\|_2 \cdot w_{\text{vel}} \quad (1)$$

$$h(s) = \|\mathbf{s}_2.\text{pos} - \mathbf{s}_1.\text{pos}\|_2 \cdot w_{\text{pos}} + \|\mathbf{s}_2.\text{vel} - \mathbf{s}_1.\text{vel}\|_2 \cdot w_{\text{vel}} \quad (2)$$

The weighted A* is a faster modification of A* that guarantees ϵ -sub-optimality, where ϵ is the weight given to the heuristic. The weight used in this paper is 1.7.

Additionally, this paper's implementation incorporates kinodynamic constraints into its re-planning framework. These constraints ensure that the transitions between states adhere to the physical and dynamic limitations of the system. Metrics such as stance time, accelerations, and ground reaction forces (GRFs) are calculated to validate the feasibility of motion between states. This integration guarantees that the paths generated during re-planning are not only efficient but also physically realizable, which is critical for applications such as quadruped locomotion or other systems requiring adherence to dynamic constraints. The plan generated by the planner can be seen in fig. 5.

Run	Nodes Expanded	Time Taken (s)	Path Cost
1	44	0.005	9.004
2	21	0.004	8.730
3	14	0.006	8.256

TABLE II: Performance of weighted A* Algorithm

C. Z-axis planning

One of the main objectives was to add a third dimension, z (height), to the planner to enable height-aware navigation. However, instead of directly sampling z, which would significantly increase the state space size and planning complexity, a 'lazy' approach has been implemented to improve the planning speed. The way this is done is by first checking if the nominal z of the robot collides with obstacles for a given x and y location. If it clears the obstacle check, that is, no collision is detected, z is left unchanged. However, if it collides, the minimum z

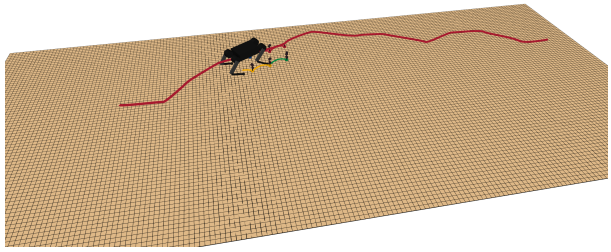
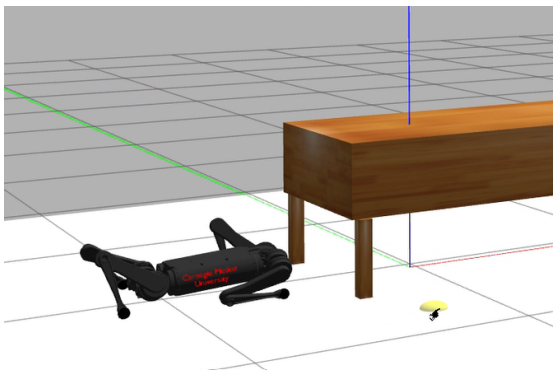
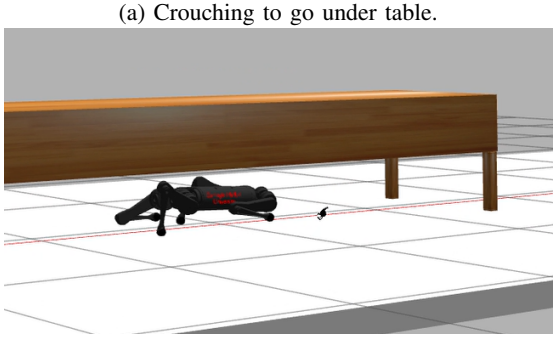


Fig. 5: Plan generated by PRM-A*

(the lowest height at which the robot can successfully walk) is checked against the obstacle. If it passes the check, the robot height is set to z minimum, effectively allowing crouching. But if even that fails, it means that the state is invalid, as the robot cannot fit through it despite crouching.



(a) Crouching to go under table.



(b) Traversing under the table

Fig. 6: Planning with z incorporated

This implementation ensures that height-based considerations are incorporated directly into the planning process, removing the reliance on post-processing. This not only improves the planner's computational efficiency but also enhances its real-time adaptability in complex environments.

D. Re-planning

In the existing implementation of the planner, which is RRT-Connect, re-planning is done continuously. However, the choice of whether to actually use the re-planned path depends on whether the new plan is significantly better than the current one or not. The current implementation checks the path quality (i.e, path length) of the new plan proposed by replanning. This makes sense for a static environment. However, since the existing planner does not consider dynamic obstacles, this metric does not account for collisions, as it is assumed that the plans generated are collision free.

Hence, the metric to validate the new path generated through the replanning has been updated to include collision checks which can handle both static and dynamic obstacles. The new plan is only used if it is collision-free, else it continues on with the older one.

E. Shortcomings

While the proposed system demonstrates significant improvements in dynamic obstacle avoidance and height-aware planning, several limitations persist that highlight areas for future refinement:

- **Search Algorithm:**
  A* and wA* are powerful and straightforward search algorithms. They lack efficiency for dynamic real-time applications as they recompute the entire path from scratch. This recomputation leads to significant computational overhead, especially when the system changes frequently. Implementing dynamic variants of A*, such as D* Lite, is recommended.
- **Computational Overhead:**
  The integration of Lazy PRM and wA* reduces the need for complete re-planning. However, real-time performance in dense, highly dynamic environments remains constrained.
- **Bounding Box Approximation:**
  Using bounding boxes for collision checking is computationally efficient. It may result in overly conservative or overly permissive planning in certain scenarios. Fine-tuning bounding box dimensions for diverse environments remains a challenge, potentially causing suboptimal paths or unexpected collisions.
- **Dynamic Obstacle Prediction:**
  The sliding window mechanism predicts the trajectories of moving obstacles. Its accuracy is limited by the precision of velocity and trajectory estimations.

These shortcomings underline the need for ongoing research and optimization to further enhance the adaptability, efficiency, and robustness of the proposed planning framework.

V. Results

The results demonstrate that while PRM wA* requires a higher number of samples and generates larger path sizes compared to RRT-Connect, it achieves significantly reduced planning times. This highlights its suitability for real-time applications where rapid adaptability and computational efficiency are prioritized over strict path optimality.

The links to the demonstration videos of the working of the implemented planner are Dynamic obstacle avoidance and Z planning incorporated.

The code is available here

Planner	Number of Samples	Path Size(m)	Planning Time
RRT-Connect	8-12	6-8	0.01
PRM A*	20000	14	0.123
PRM wA*	20000	16	0.006

TABLE III: PRM A* vs PRM wA* vs RRT-Connect

VI. Conclusion

This work represents an advancement in the planning of quadruped robots for dynamic environments in the Quad-SDK framework. By integrating Lazy PRM with weighted A*, the proposed system efficiently handles dynamic obstacle avoidance and incorporates Z-axis planning without the need for post-processing. The lazy approach to Z-dimension planning successfully reduces state space complexity while maintaining robust adaptability in real-time. This enables the robot to dynamically adjust its height, such as crouching, to navigate through challenging terrains.

Despite these enhancements, the system encounters limitations when operating in environments with dense and highly dynamic obstacles. The increased frequency of re-planning and the growing state-space complexity in such scenarios can pose significant computational challenges. These issues may affect the system's ability to maintain real-time performance under extreme conditions. Future work will need to focus on optimizing the framework to better handle these scenarios, ensuring scalability and robust adaptability for deployment in more demanding and complex environments.

Acknowledgment

The authors would like to thank the creators of Quad-SDK, and the instructors/teaching assistants of 16-782 for their insights.

References

[1] Robert Bohlin and Lydia E. Kavraki. "Path Planning Using Lazy PRM". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2000, pp. 521–528.

[2] Dave Ferguson and Anthony Stentz. "Anytime RRTs". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2006, pp. 5369–5375.

[3] Sven Koenig and Maxim Likhachev. "D* Lite". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2002, pp. 476–483.

[4] SeungJong Noh, Daeyoung Shim, and Moongu Jeon. "Adaptive Sliding-Window Strategy for Vehicle Detection in Highway Environments". In: *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. IEEE, 2015, pp. 660–665. DOI: 10.1109/ITSC.2015.125. URL: https://ieeexplore.ieee.org/document/7226849.

[5] Joseph Norby and Aaron M. Johnson. "Fast global motion planning for dynamic legged robots". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 3829–3836.

[6] Joseph Norby et al. "Quad-SDK: Full Stack Software Framework for Agile Quadrupedal Locomotion". In: *ICRA Workshop on Legged Robots*. May 2022. URL: https://leggedrobots.org/index.html.