Parallel Motion Planning with Neural Surrogate Models

Leo Bashaw
Department of Electrical and Computer Engineering
Rice University
Houston, TX
lb73@rice.edu

***Abstract*—Businesses in healthcare, defense, and heavy industry are becoming increasingly reliant on autonomous robots to complete a variety of tasks. To maximize their effectiveness, these robots must be able to replan trajectories in real-time as their environments change. Therefore, a high-speed planning pipeline that exploits modern GPU architectures is desirable. Moreover, in industries that leverage human-in-the-loop robotics (e.g. healthcare), solution paths must be more than smooth or efficient, they must also satisfy nontraditional metrics (e.g. safety). There is a lack of open source, real-time planning libraries that generate solution paths optimized for qualities other than traditional metrics such as smoothness or energy efficiency. We propose a GPU-enabled roadmap-based planner with an embedded neural surrogate model to score graph nodes on non-traditional metrics. Solution paths are generated by a search algorithm that incorporates these scores into its heuristic function. We aim to provide open source software that generates paths which maximize the user's learned cost-function, and does so at real-time frequencies.**

***Index Terms*—Perception-aware planning, parallel planning, neural networks.**

I. Introduction

Motion planning for robotic systems in dynamic environments presents a fundamental challenge that must be addressed to ensure their safe autonomous operation. As robots become increasingly integrated into industrial environments, the demand for sophisticated planning capabilities has grown rapidly. Businesses require algorithms that can run in real time, as well as solutions that are optimized for domain-specific metrics beyond standard efficiency measures.

Recent publications on real-time motion planning study the problem within the frame of autonomous driving [12]. Essentially all of these systems leverage modern GPU hardware to achieve real-time performance, and the use of GPUs (or more generally multithreaded computing) to accelerate motion planning algorithms has been around since the early 2000s [2], [6], [8], [9]. Three traditional approaches to motion planning- roadmap, tree, and optimization-based algorithms- possess varying degrees of parallelism. The sequential nature of tree construction and iterative optimization processes prevents them from possessing the massive parallelism that is inherent in roadmap construction, though there are successful GPU-based implementations of each type of algorithm [2], [6], [10].

The proliferation of GPU hardware in modern computing environments offers an opportunity to radically accelerate planning algorithms that have historically been constrained by computational limitations. Moreover, the accessibility of neural networks and the speed at which inference can be run provide an opportunity to embed networks in real-time planning pipelines. Specifically for perception-aware planning, deep learning is used to detect and classify objects, segment environments, and steer planning algorithms towards texture-rich environments in an effort to minimize localization uncertainty [1], [5], [13]. However, we feel there is a lack of research on perception-aware real-time motion planners that generate solution paths for maximal visibility of some object in the environment (e.g. an enemy soldier in a combat environment or a delicate body part during surgery).

Our project addresses this technological gap by developing a GPU-accelerated planning framework that combines the strengths of probabilistic roadmaps (PRM) with neural surrogate models [7]. We aim to achieve planning frequencies that support real-time re-planning in dynamic environments by leveraging the capabilities of modern GPUs. Our approach incorporates learned cost functions (via embedded neural surrogate models) that assign scores to roadmap nodes that measure non-traditional metrics such as privacy or safety. This approach enables the generation of solution paths optimized for both traditional metrics (such as path length or energy efficiency) and the learned metrics described above.

This report outlines our progress toward creating an open-source planning library that can generate high-quality motion plans optimized for learned metrics at real-time speeds. We detail our implementation approach, current achievements, and discuss our plan to implement the remaining components of our system. The ultimate goal is to provide the robotics community with a versatile tool that can adapt to diverse application requirements while maintaining a performance level suitable for real-world deployment.

II. Methods

To build our parallelized planner, we will leverage several libraries that perform graph analytics, neural inference, and collision checking on GPUs - we make extensive use of CuRobo from Nvidia [11]. Moreover, we will write our own kernels where necessary to handle operations such as state

generation and construction of graph edges. Figure 1 shows a flowchart of our pipeline. Notice that the neural model which provides scores for each node can be swapped out according to the use case, that is, we can use different networks that have been trained to learn different types of scores. This modularity is highly beneficial because it allows the user to take advantage of our pipeline without being limited to the applications that we focus on. The inference kernel can also be run in parallel with the edge-construction kernel, suggesting that this added functionality will come at little extra computational cost.

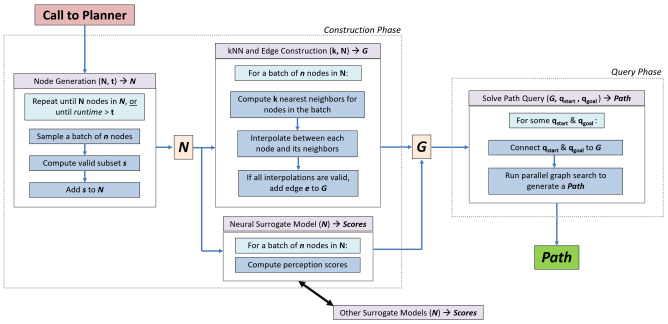

Fig. 1. The parallel planning and surrogate model pipeline.

To date, we have written the state generation kernel and benchmarked it against the state sampling function provided in CuRobo. This benchmarking was run across a range of batch sizes for 10000 trials per batch size. We also benchmarked our inference kernel (built with TensorRT) against the inference functions provided in Pytorch. Again, we benchmarked across a range of batch sizes for 10000 trials each. All benchmarking was performed on a laptop with an RTX 3050 Ti and an Intel i9 12900-HK. Once this project is complete, we will run benchmarking on a workstation equipped with an RTX 4090 and a more powerful CPU.

III. Results

Our initial benchmarking results demonstrate promising performance gains in two key components of our planning framework: state generation and node scoring with the neural model. Since these are the only components that we have implemented and benchmarked so far, the data presented and the subsequent discussion will focus only on these.

A. State Generation

Figure 2 shows box-plots of the runtime for our state-generation kernel compared to the function provided in CuRobo. Notice that for smaller batch sizes (at or below 10,000), our implementation is faster than CuRobo's by roughly a factor of 2. We also achieve a tighter IQR across all batch sizes, which suggests that we achieve more predictable performance. That being said, our kernel suffers from scalability issues in that run times rise in a non-linear fashion as batch size increases, while CuRobo is far more stable.

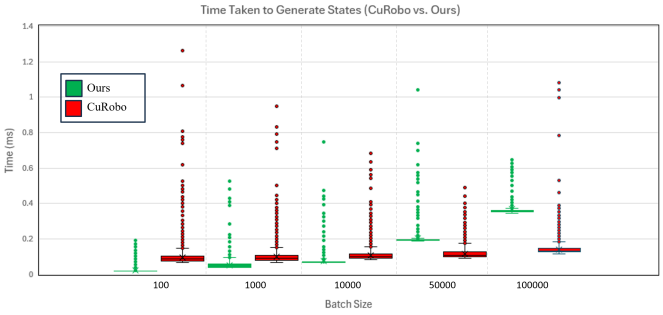

Fig. 2. State generation kernel performance statistics.

B. Node Scoring

Figure 3 shows the mean runtime of our inference kernel when we build our model in Pytorch and on TensorRT. The TensorRT implementation of the kernel is significantly faster than standard Pytorch inference with a speedup factor of 2-3 depending on the batch size. Note that inference times start to climb exponentially around batch sizes of 20,000. Figure 4 shows the standard deviations for the same data, and again we see the benefits of implementing our kernel with TensorRT. The standard deviations are much lower across the range of batch sizes, suggesting not only better but also more predictable performance.

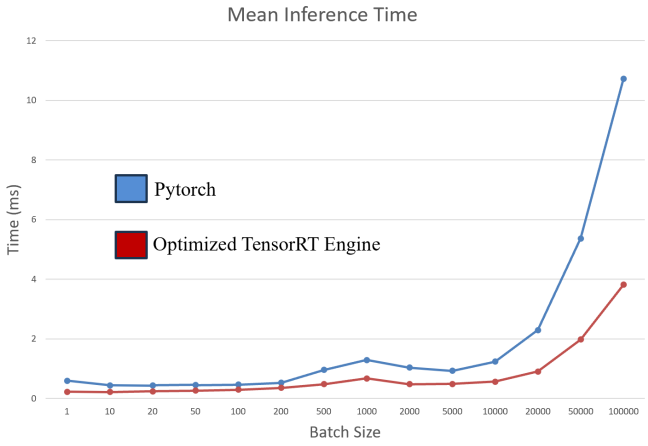

Fig. 3. Mean runtime for surrogate model inference.

IV. Discussion

A. State Generation Performance Analysis

The superior performance of our state generation kernel for smaller batch sizes demonstrates an important advantage for real-time planning scenarios. In dynamic environments where rapid re-planning is essential, we may prefer to use smaller batch sizes to minimize latency. Our implementation's ability to generate states twice as efficiently in these ranges directly contributes to reducing overall planning time. The tighter IQR observed across all batch sizes is particularly valuable in real-time systems where predictable performance is often as

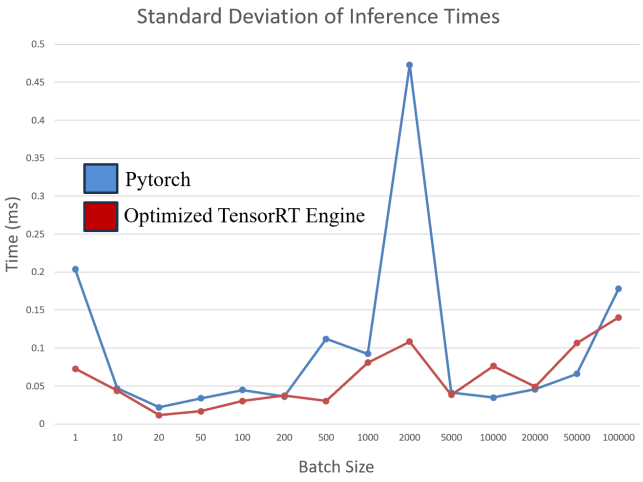

Fig. 4. Standard deviations of runtime for surrogate model inference.

important as raw speed. Consistent execution times allow for more reliable planning cycles and better temporal guarantees for the algorithm as a whole.

However, the non-linear scaling behavior at larger batch sizes presents a significant challenge for applications requiring extensive configuration space exploration. This scalability limitation suggests that our kernel implementation may benefit from architectural refinements:

- Memory access patterns might be causing issues for larger batch sizes.
- Thread block configuration may need optimization to better balance parallelism and resource utilization.
- Kernel launch overhead might be amplified by our current batching strategy.

Given these characteristics, our planning framework will likely perform best when operating with multiple smaller batches rather than fewer large ones.

B. Neural Surrogate Model Performance

The 2-3x performance advantage of our TensorRT-optimized inference engine validates our approach to integrating learned cost functions into the planning pipeline. This substantial speed-up allows complex neural models to be incorporated without compromising real-time performance constraints.

The exponential increase in inference time beyond batch sizes of 20,000 for both implementations highlights an important constraint for our planning framework. This performance characteristic establishes a practical upper bound for batch processing and reinforces the need for intelligent workload partitioning in the complete system.

The significantly lower standard deviations achieved by the TensorRT implementation represent perhaps an even more important advantage than the raw speed improvements. For real-time robotic systems, especially those operating in safety-critical environments, the predictability of execution times directly impacts reliability. Lower variability enables tighter timing guarantees and reduces the need for conservative planning horizons to accommodate worst-case scenarios.

C. Implications for Integration

The benchmarking results from both components provide some valuable initial guidance as we work towards a fully integrated pipeline:

- Batch sizes should be kept around 10,000 to maximize the performance advantages of our state generation kernel while avoiding the exponential performance degradation during surrogate model inference.
- The included benchmarks were performed on a laptop-grade GPU. The planned deployment on an RTX 4090 workstation is likely to yield substantially better performance, potentially enabling higher batch sizes without encountering the same scaling limitations.
- The complementary strengths of our state generation and neural inference kernels align well with our goal of creating a real-time planning framework that incorporates learned path optimization metrics.

These initial results validate our fundamental approach, though there is still much work to do, and we will likely encounter unexpected overhead once we start integrating kernels. We see a clear path forward from this checkpoint: implement the remaining kernels and continually test their integration. It will be crucial to continue benchmarking performance as we move forward to avoid unexpected issues once we begin testing our full planning pipeline.

V. Conclusion

This report presents our work to date on a parallelized roadmap-based planner that can run at high frequencies in dynamic environments that require re-planning. This planner will have perception scores associated with each node that are generated by a neural network. Moving forward, we intend to finish implementing all kernels in our pipeline, complete the integration and debugging in simulation, and move to optimizing hyper-parameters to achieve a performance goal of less than 50 ms for a complete run. We feel that this research has a strong link to the medical and defense industries, since both fields demand autonomous monitoring of certain objects while other tasks are completed.

VI. Acknowledgment

I would like to thank my advisors at Rice University, Lydia Kavraki and Jose Moreto, for their continued support and feedback. I also want to thank the researchers and engineers at Nvidia, whose work on CuRobo and GPUs made this research possible. Finally, I want to acknowledge the creators of the Rapids AI ecosystem for their excellent parallel graph analytics library.

References

[1] L. Bartolomei, L. Teixeira, and M. Chli, "Perception-aware Path Planning for UAVs using Semantic Segmentation," in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Oct. 2020, pp. 5808–5815. doi: 10.1109/IROS45743.2020.9341347.

[2] J. Blankenburg, R. Kelley, D. Feil-Seifer, R. Wu, L. Barford, and F. C. Harris, "Towards GPU-Accelerated PRM for Autonomous Navigation," in 17th International Conference on Information Technology–New Generations (ITNG 2020), S. Latifi, Ed., Cham: Springer International Publishing, 2020, pp. 563–569. doi: 10.1007/978-3-030-43020-774.

[3] M. Campana, F. Lamiraux, and J.-P. Laumond, "A gradient-based path optimization method for motion planning," Advanced Robotics, vol. 30, no. 17–18, pp. 1126–1144, Sep. 2016, doi: 10.1080/01691864.2016.1168317.

[4] J. Cortés and T. Siméon, "Sampling-Based Tree Planners (RRT, EST, and Variations)," in Encyclopedia of Robotics, M. H. Ang, O. Khatib, and B. Siciliano, Eds., Berlin, Heidelberg: Springer, 2020, pp. 1–9. doi: 10.1007/978-3-642-41610-1170 − 1.

[5] G. Costante, C. Forster, J. Delmerico, P. Valigi, and D. Scaramuzza, "Perception-aware Path Planning," Feb. 10, 2017, arXiv: arXiv:1605.04151. doi: 10.48550/arXiv.1605.04151.

[6] C. H. Huang, P. Jadhav, B. Plancher, and Z. Kingston, "pRRTC: GPU-Parallel RRT-Connect for Fast, Consistent, and Low-Cost Motion Planning," Mar. 09, 2025, arXiv: arXiv:2503.06757. doi: 10.48550/arXiv.2503.06757.

[7] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," IEEE Transactions on Robotics and Automation, vol. 12, no. 4, pp. 566–580, Aug. 1996, doi: 10.1109/70.508439.

[8] J. Pan, C. Lauterbach, and D. Manocha, "g-Planner: Real-time Motion Planning and Global Navigation using GPUs," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 24, no. 1, Art. no. 1, Jul. 2010, doi: 10.1609/aaai.v24i1.7732.

[9] E. Plaku, K. E. Bekris, B. Y. Chen, A. M. Ladd, and L. E. Kavraki, "Sampling-based roadmap of trees for parallel motion planning," IEEE Transactions on Robotics, vol. 21, no. 4, pp. 597–608, Aug. 2005, doi: 10.1109/TRO.2005.847599.

[10] D. Shah, N. Yang, and T. M. Aamodt, "Energy-Efficient Realtime Motion Planning," in Proceedings of the 50th Annual International Symposium on Computer Architecture, in ISCA '23. New York, NY, USA: Association for Computing Machinery, Jun. 2023, pp. 1–17. doi: 10.1145/3579371.3589092.

[11] B. Sundaralingam et al., "cuRobo: Parallelized Collision-Free Minimum-Jerk Robot Motion Generation," Nov. 03, 2023, arXiv: arXiv:2310.17274. doi: 10.48550/arXiv.2310.17274.

[12] S. Teng et al., "Motion Planning for Autonomous Driving: The State of the Art and Future Perspectives," IEEE Transactions on Intelligent Vehicles, vol. 8, no. 6, pp. 3692–3711, Jun. 2023, doi: 10.1109/TIV.2023.3274536.

[13] Y. Zhao, Z. Xiong, S. Zhou, J. Wang, L. Zhang, and P. Campoy, "Perception-Aware Planning for Active SLAM in Dynamic Environments," Remote Sensing, vol. 14, no. 11, Art. no. 11, Jan. 2022, doi: 10.3390/rs14112584.