RAPIDLY-EXPLORING RANDOM TREES SEARCH ALGORITHM WITH GRAPH NEURAL NETWORKS FOR SAMPLE-BASED PATH PLANNING

Postgraduate Research Proposal

Mughees Asif

28th March 2022

Abstract

Autonomous motion planning for robotic systems in rich topological environments requires advanced route navigation and evaluation of path suitability. Recent trends have seen the integration of sample-based planning algorithms such as Rapidly-exploring Random Trees (RRT) with deep learning frameworks including Convolutional Neural Networks (CNNs) to develop path-planning control systems capable of finding a collision-free path from an origin to a destination. This research proposal suggests the interfacing of advanced Robotics with Artificial Intelligence by using Graph Neural Networks with RRTs to enable high-level spatial perception that minimises the loss of any information about the environment whilst preserving navigational stability.

Introduction

Motion planning has been researched widely to enable robotic systems in domains such as trajectory planning, coordinating several robots simultaneously, and handling kinodynamic limitations in unpredictable state space environments [0]. Baseline searching algorithms include Dijkstra's that uses incremental division of the geometry of the whole state space into finite elements for analysis, and A* search that augments with Dijkstra's by using a heuristic to *inform* the search [1]. Rapidly-exploring Random Trees (RRT) search improves on both by introducing Sample-based Planning (SBP) that allows the robotic system to approximate the topology of all it's possible configurations within the configuration space (C-space) [2]. The algorithm also guarantees theoretical probabilistic completeness as all points in the C-space are sampled and connected via a computational graph data structure representation, where a optimal path is always found [3].

Recently, researchers have also used deep learning (DL) with SBP to extract the most pertinent samples in the C-space for a given task [2]. These samples are established when deciding which features are important for the robotic system's kinematics and, thereby, due to the vast computational capacity of DL, the methodology offsets the 'curse of dimensionality' that ensures a stable increase in the C-space dimensionality as the samples required to cover the space increase [4].

A common research trend amongst researchers is the usage of vanilla Feed-forward Neural Networks (FCNs) and Convolutional Neural Networks (CNNs) to extract the salient samples from the state space. However, FCNs and CNNs do not make for a conducive environment when enabling autonomous robotic systems in a *3D* environment, as most of the rich topological information is lost during the sampling. In this proposed research, the author will establish how using Graph Neural Networks (GNNs) with the RRT search algorithm can increase agent generalization of any given state space by representing the environment in a graphical structure to maintain topological integrity. The research aims to address two main problems using GNNs: (a) identify critical samples that are relevant to the motion planning of the system, and (b) use the critical samples to prune the search towards the fastest collision-free route towards the destination. The solution will be tested using a computational agent that has to navigate from an origin to a final destination within a maze environment.

Methodology

This section provides a high-level overview of the algorithms that will be used for the project. Additionally, the main programmatic implementation will be done using the `Python` programming language in a `Jupyter Notebook` environment for a complete end-to-end solution that will be made open-source.

Sample-based Planning (SBP)

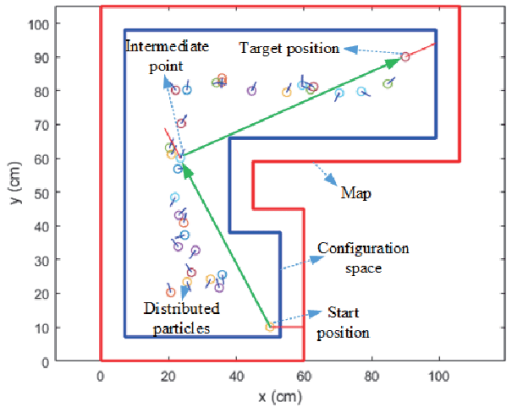

Figure 1: Sample-based planning [5]

Sample-based planning relies "... on a collision checking module, providing information about feasibility of candidate trajectories, and connect a set of points sampled from the obstacle-free space in order to build a graph (roadmap) of feasible trajectories. The roadmap is then used to construct the solution to the original motion-planning problem" [2]. Fig. 1 displays SBP for a robotic system from a start position to a target position, where, the distributed particles are possible routes, and the green line represents the optimal path. RRT builds on this technique by introducing *incremental* SBP that improves computational cost by preventing the unnecessary setting of samples as a priori i.e., building a graph with redundant connections per node. This results in probabilistic completeness where the probability of failure to find an optimal route decays to zero, exponentially.

Rapidly-exploring Random Trees (RRT)

Fig. 2 shows the RRT algorithm output for a *3D* environment with obstacles. The original path with the all the associated nodes is highlighted in blue while the green path shows the RRT path navigation. The optimized path shows the linear travelling route from the start to the goal node, and the obstacles are expressed as red spheres. RRT uses a space-filling tree that has been randomly generated from samples drawn in from the state space and gives preference to the propagation of the search towards unsearched regions [6]. As each tree develops, connections/vertices towards the nearest tree are made and, if the connection is progressive i.e., moves towards the goal and adheres to given constraints, the nearest tree is added as a *state* to the developing tree [6].

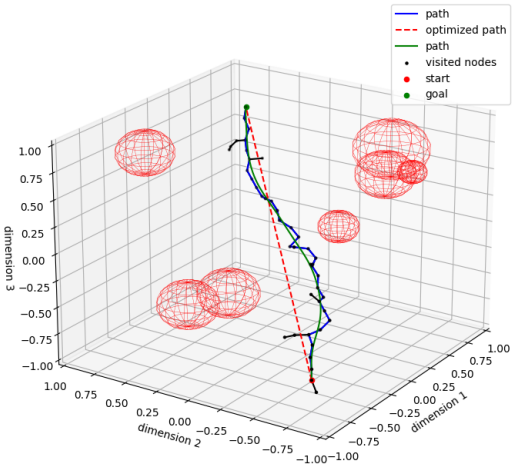

Figure 2: RRT search in a *3D* environment

Algorithm 1 Rapidly-exploring Random Trees (RRT)

Input: Initial configuration x_{init}, number of vertices K, time increment Δt
Output: RRT graph $\mathcal{T}$

$\mathcal{T}$.Init(x_{init}); ▷ Initialise graph with input parameters
for $k = 1$ to K **do**
  $x_{rand} \leftarrow$ RANDOM_STATE(); ▷ Select a random configuration
  $x_{near} \leftarrow$ NEAREST_NEIGHBOUR($x_{rand}, \mathcal{T}$); ▷ Calculate distance towards neighbour
  $u \leftarrow$ SELECT_INPUT(x_{rand}, x_{near}); ▷ Move towards neighbour
  $x_{new} \leftarrow$ NEW_STATE($x_{near}, u, \Delta t$); ▷ Establish new state at neighbouring node
  $\mathcal{T}$.add_vertex(x_{new}); ▷ Add vertex at new state
  $\mathcal{T}$.add_edge(x_{near}, x_{new}, u); ▷ Add edge towards the new state
end for
return $\mathcal{T}$

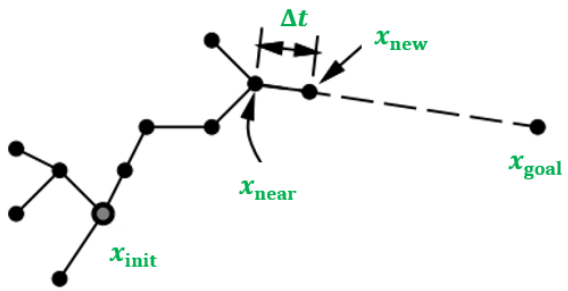

Figure 3: Selection process to establish a new state

Algorithm 1 highlights the baseline RRT algorithm, where path planning is observable as a metric space search from an initial state x_{init} to a goal x_{goal}. "A simple iteration is performed in which each step attempts to extend the RRT by adding a new vertex that is biased by a randomly-selected state" [6]. Fig. 3 displays the selection process where the RRT is extended, and the next vertex is chosen according to a distance metric ρ. The `NEW_STATE` function enables motion towards the new position x_{new} via an input u with a time increment Δt [6].

Graph Neural Networks (GNNs)

GNNs are deep learning frameworks that perform inference on data that can be represented by a set of nodes and the associating vertices. This static input graph is then manipulated through various layers of convolution filters to aggregate information from the nodes [7]. The relationship between data x and a graph $\mathcal{G} = \{V, E\}$ consisting of N nodes V and edges $E \subseteq V \times V$, can be characterised by the matrix shift operator $\mathbf{S}$ [8]. The aggregation operation $[\mathbf{Sx}]_n$ for a set of neighbouring nodes n denoted by $\mathcal{B}_n$ can thus be defined as,

$$[\mathbf{Sx}]_n = \sum_{j=n, j \in \mathcal{B}_n} s_{nj} x_n \qquad (1)$$

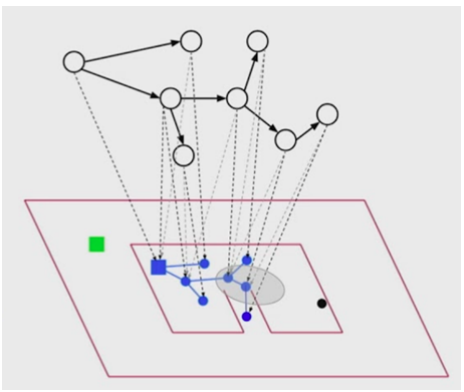

Figure 4: Graph representation of path planning

If the neighbouring nodes are accessed recursively, then graph convolution can be described as a simple repeated application of a filter to an input that results in a feature map that in indicative of the locations and strengths of a detected

feature in an input [7],

$$[\mathbf{z}] = \sum_{k=0}^{K} h_k \mathbf{S}^k \mathbf{x} = \mathbf{H}(\mathbf{S})\mathbf{x} \tag{2}$$

where,

$$\mathbf{H}(\mathbf{S}) = \sum_{k=0}^{K} h_k \mathbf{S}^k$$

that represents a linear shift invariant graph filter with coefficients h_k [8]. The output of the layer is adjusted for nonlinearity σ (using the ReLU activation function) and derived as,

$$\mathbf{y_1} = \sigma[\mathbf{z_1}] = \sigma\left[\sum_{k=0}^{K} h_{1k} \mathbf{S}^k \mathbf{x}\right] \tag{3}$$

where, when applied recursively on L layers results in,

$$\mathbf{y_L} = \sigma[\mathbf{z_L}] = \sigma\left[\sum_{k=0}^{K} h_{Lk} \mathbf{S}^k \mathbf{y_{L-1}}\right] \tag{4}$$

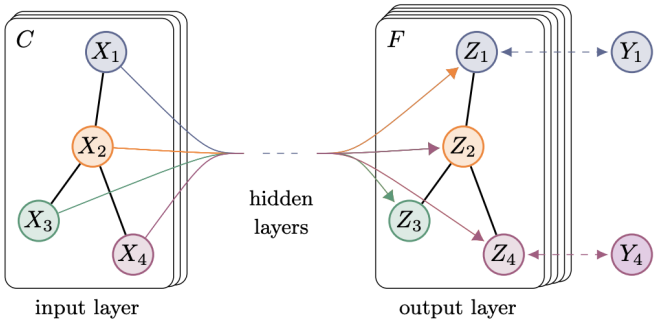

Figure 5: GNN input and output layers [9]

Therefore, Eq. 3 & 4 represent the main graph convolutions (Fig. 5) required to extract the relevant information about a given trajectory, where each convolutional layer uses integrals to express the amount of overlap faced by each feature, as it is shifted over to the next node [9].

Timeline

Timeline 1: Proposed timeline for the research (2022)

February	Preliminary research
March	Research proposal development and discussion with supervisor
April	Draft up and test computational implementations
May	Tune and test final implementation
June	Tune and test final implementation
July	Write up the corresponding report
August	Submission (Report, Presentation & Viva)

Conclusion

The aim of this research is to develop a high-level motion planning system that is capable of path detection and navigation. The main techniques that have been proposed include sample-based planning in the form of the Rapidly-exploring Random Trees search algorithm that leverages Graph Neural Networks to decipher a collision-free path and allows the robotic system to develop rich topological information about any given environment. The proposed advantages of this methodology include increased spatial generalization for different environmental geometries and enabling high-level decision-making skills for increased kinodynamic autonomy.

References

[0] J.-C. Latombe, "Motion planning: A journey of robots, molecules, digital actors, and other artifacts," *The International Journal of Robotics Research*, vol. 18, no. 11, pp. 1119–1128, 1999. DOI: 10.1177/02783649922067753. eprint: https://doi.org/10.1177/02783649922067753. [Online]. Available: https://doi.org/10.1177/02783649922067753.

[1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall, 2010.

[2] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011. DOI: 10.1177/0278364911406761. eprint: https://doi.org/10.1177/0278364911406761. [Online]. Available: https://doi.org/10.1177/0278364911406761.

[3] M. Kleinbort, K. Solovey, Z. Littlefield, K. E. Bekris, and D. Halperin, "Probabilistic completeness of rrt for geometric and kinodynamic planning with forward propagation," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. x–xvi, 2019. DOI: 10.1109/LRA.2018.2888947.

[4] D. Hsu, J.-c. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," *International Journal of Computational Geometry & Applications*, vol. 09, Mar. 1997. DOI: 10.1142/S0218195999000285.

[5] Y. Liu, B. Yu, R. Fan, M. Liu, M. Bocus, H. Ni, J. Fan, and S. Mao, "Mobile robot localisation and navigation using lego nxt and ultrasonic sensor," Dec. 2018, pp. 1088–1093. DOI: 10.1109/ROBIO.2018.8665350.

[6] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Iowa State University, Tech. Rep., 1998.

[7] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, *Graph neural networks: A review of methods and applications*, 2018. DOI: 10.48550/ARXIV.1812.08434. [Online]. Available: https://arxiv.org/abs/1812.08434.

[8] A. Khan, A. Ribeiro, V. R. Kumar, and A. G. Francis, "Graph neural networks for motion planning," *ArXiv*, vol. abs/2006.06248, 2020.

[9] T. N. Kipf and M. Welling, *Semi-supervised classification with graph convolutional networks*, 2016. DOI: 10.48550/ARXIV.1609.02907. [Online]. Available: https://arxiv.org/abs/1609.02907.