TETRAHEDRAL SOFT ROBOT NAVIGATION

FINAL PROJECT

MEEN 689 – Planning and Controlling of Autonomous Vehicle

Perera, Dulanjana
dperera@tamu.edu

1 Introduction

Soft Robotics is an emerging field with great potential in many applications, such as pick-and-place delicate objects, search-and-rescue operations, and non-invasive brain surgeries [1]. However, due to its stochastic nature, modeling and controlling are challenging. Hence, soft mobile robot navigation needs further exploration to perform simple navigation tasks without human intervention.

Numerous methods have been introduced for the navigation of mobile robots, especially for wheel and aerial robots. The navigation can be divided into two sections based on the objectives [2]. The Planning phase plans the path for given constraints, and the Controlling phase ensures that the robot follows the generated path from the planning phase. The most common graph-based optimum planning methods are A* [3] and Dijkstra [4], which apply brute force to search the path while avoiding obstacles. It can be applied to both local and global planning applications. However, these methods are computationally inefficient and not suitable for real-time applications. The other popular methods are Genetic algorithms [5], Reinforcement Learning methods (path and Policy) [6], and the Artificial potential field [7] approach. RL and Artificial potential field methods can be applied globally, but these fail to generalize the planning. A few online planning approaches have been introduced, such as visibility graphs and cell decomposition road maps (verticle, Voronoi, and grid) [8]. The generated map is suboptimum and cannot guarantee that a path can be generated for any given location. The sample-based road-map approaches can generate a path that is suboptimum. For example, RRT, RRT* [9], and probabilistic road maps [10]. These are base versions of path planning, and modified versions have been introduced for online (on-the-fly) planning for autonomous vehicles.

Once the path is generated, the controller ensures that the vehicle follows the path with a defined error [11]. In practice, no algorithm can control the vehicle to follow the exact path due to the uncertainties associated with the robot and the environment. Therefore, modern controllers are developed to compensate for uncertainties and handle unforeseen dynamics. The earliest methods of controlling vehicles are PID and pure pursuit approaches. These are not robust enough to handle unanticipated changes in the path or the environment. The state space controllers (Reference tracking and pole placement) can accommodate uncertainties to a certain level but are not very robust for high-frequency applications (racing cars).

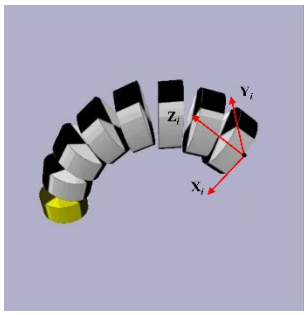
Moreover, value iteration and LQR have not been developed to handle these changes [12]. However, LQG can account for noise in the system and environment. MPC and sliding mode control methods have been introduced to address these issues [13]. The MPC can predict the future (see the future) and decide the control sequence to follow the future path segments while limiting the system's property (fuel cost or time). Therefore, dynamic changes are captured and handled by the algorithm.

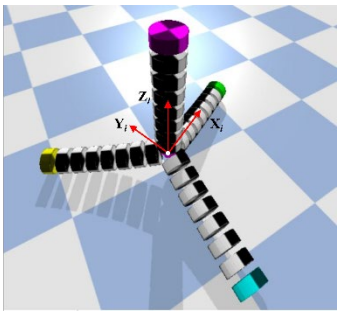
Soft robots are, in general, highly nonlinear systems. Therefore, deriving a kinematic/dynamic model for navigation is very challenging. Besides, based on the actuation mechanism (pneumatic, cable, or rod), and the

locomotion gait of the robot (crawling, trotting, rolling, etc.), predicting the next state incorporates uncertainty which can cause undesired results. Therefore, planning and controlling need sophisticated algorithms. Moreover, approximations and assumptions must be considered to apply the abovementioned algorithms. To my knowledge, only one autonomous navigation scheme has been proposed for soft mobile robots [14], and it employs a simple algorithm that is similar to pure pursuit. The algorithm performs left and right turns to keep the robot within an error boundary. Apart from that, there are no autonomous control schemes for soft-legged robots.

The main reason for scarcity in the autonomous navigation system for soft mobile robots is the difficulty of modeling the transition function associated with uncertainty. Also, most of the research still develops mobile soft robots that are suitable for real-world deployments. In this project, I apply the value iteration algorithm for a finite horizon and obtain the policy for each horizon. The following section discusses the problem formulation and the algorithm in detail.

2 Problem formulation





A demand is there to explore suitable planning and control methods for autonomous navigation for soft robots. This project uses a soft tetrahedral robot and the PyBullet simulation environment (Physics-based) to apply the algorithm [14]. The robot has four continuum legs that are bent in a constant curvature arc. Due to the tetrahedral joint, the system shows stability and symetricity. The rolling gait is employed as the locomotion type. An approximated model is proposed for the complete-robot locomotion because rolling gait populates honeycomb-like discrete space for task space. The actions space only contains three actions at a given time. However, due to the stochastic nature of the robot (even in the simulation), simulation is used as the transition for the algorithm.

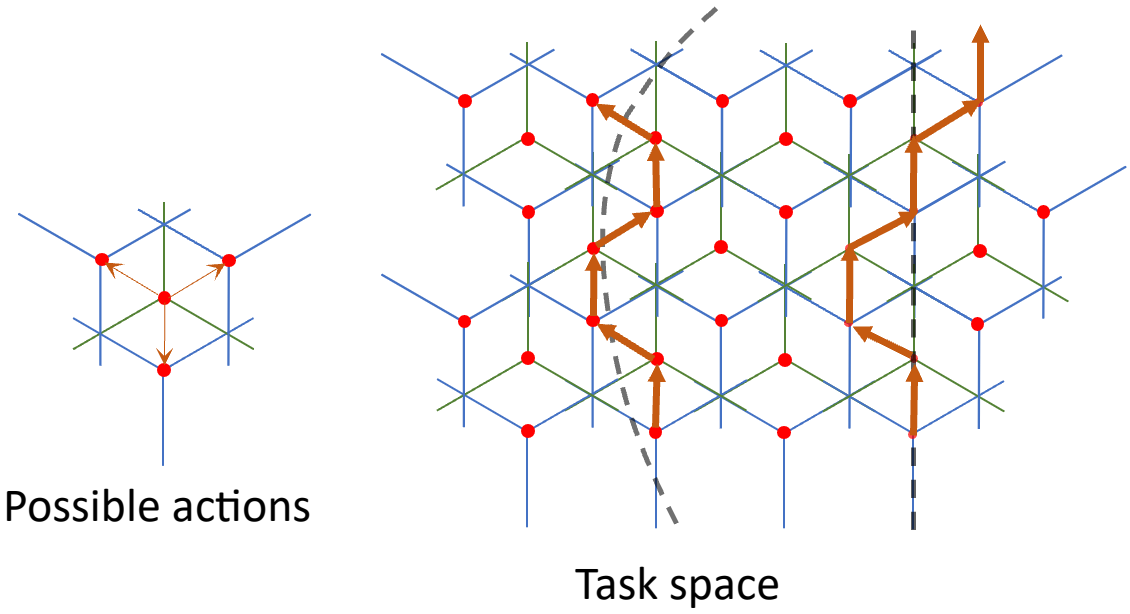


Figure 1: Discrete Honecomb cartesian space

The system is formulated as 3 States, which include global x, global y, and the orientation of the robot. The interesting fact is due to the tetrahedral shape, the robot can only exist in 8 different orientations.

- Global X ∈ ℝ
- Global Y ∈ ℝ
- Orientation of the robot $O \in \mathbb{R}^3$: $[-\pi, \pi]$

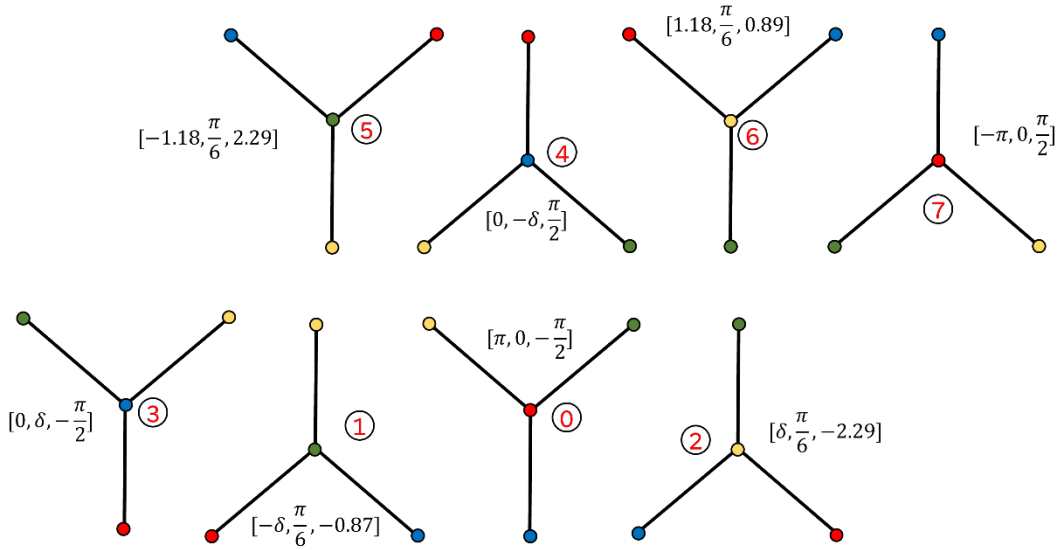


Figure 2: Orientation with respect to global frame and orientation index for the algorithm

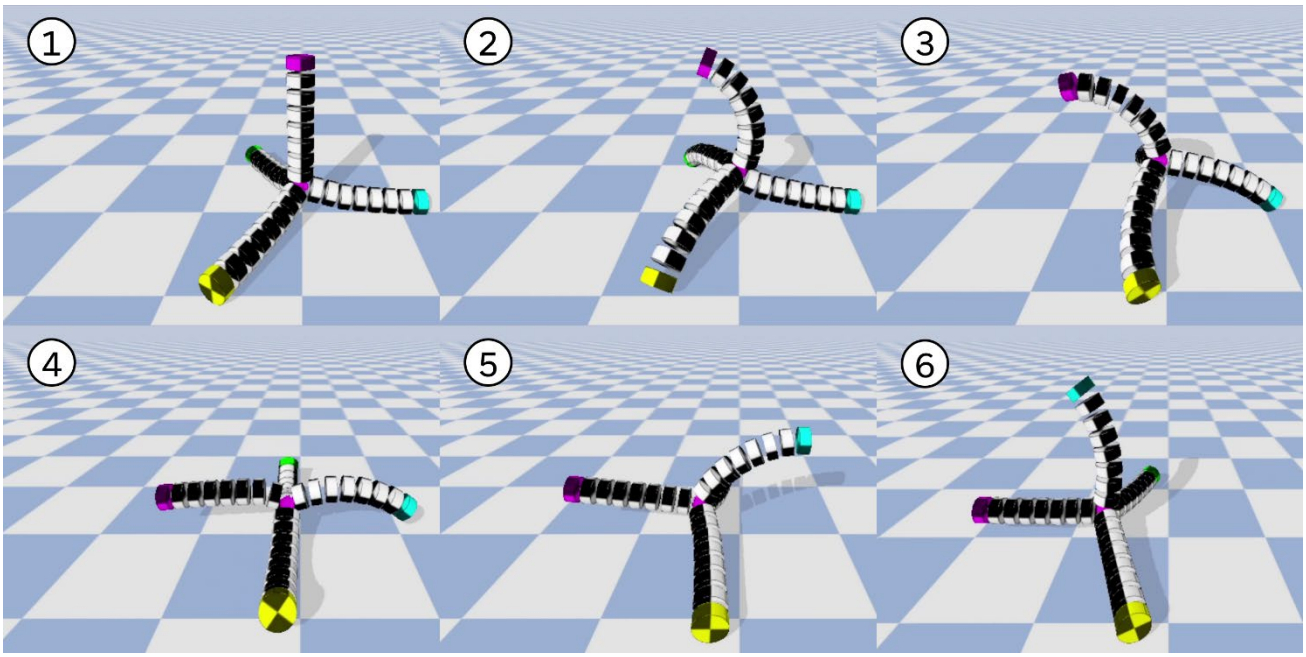
As the figure illustrates, each leg has two possibilities of being the head, and the orientation is rotated around the *z-axis* π. These orientations are considered when developing the algorithm and the simulation environment for the robot.

The action is dependent on the head. Each head position has 3 directions to roll.

- $blue : \{'red', 'green', 'yellow'\}$
- $red : \{'blue', 'green', 'yellow'\}$
- $green : \{'red', 'blue', 'yellow'\}$
- $yellow : \{'blue', 'red', 'green'\}$

2.1 Action:

The input to the system is a string that describes the tail. The system automatically identifies the head and performs corresponding rolling action. The rolling is predefined in the system. The following figure illustrates the rolling sequence.

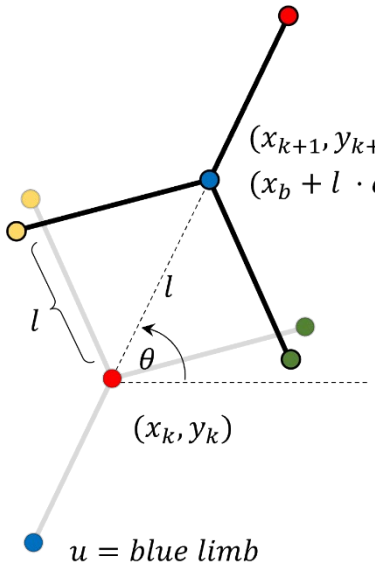


2.2 Descritization:

As mentioned in the problem formulation, the position of the robot is on the points of the honeycomb structure. Therefore, cartesian space is discrete. However, for the value iteration, the state space is discretized with 2cm resolution to have a considerable grid size. The orientation is, by default discrete due to the tetrahedron shape.

2.3 State Transition Model

The derivation of the model assumes that rolling only happens in the direction of the tail. With that, the following geometrical derivation for the next state can be obtained.



$$\theta = \cos^{-1}\left(\frac{(x_k - f_x(u))}{\sqrt{(x_k - f_x(u))^2 + (y_k - f_y(u))^2}}\right)$$

$f_x(u): \mathbb{N} \rightarrow \mathbb{R}$; $current\ x\ coodinate\ of\ the\ tip\ of\ the\ u^{th} limb$

$f_y(u): \mathbb{N} \rightarrow \mathbb{R}$; $current\ y\ coodinate\ of\ the\ tip\ of\ the\ u^{th} limb$

Here, $f_x(u), f_y(u), f_o(u)$ are the sensor functions. The environment has all the states available to read.

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ O_{k+1} \end{bmatrix} = \begin{vmatrix} x_k + \dfrac{l \cdot (x_k - f_x(u))}{\sqrt{(x_k - f_x(u))^2 + (y_k - f_y(u))^2}} \\ y_k + l \cdot \sqrt{1 - \dfrac{(x_k - f_x(u))^2}{(x_k - f_x(u))^2 + (y_k - f_y(u))^2}} \\ f_o(u) \end{vmatrix}$$

2.4 Pybullet-based Environment

The model of the robot is in 'urdf' format and it is loaded to the environment. The Pybullet is used to simulate the physics associated with the robot, such as friction, gravity and other mechanical properties such as joint damping.

The environment was developed to read all the states, and the robot can be placed in any place at any given orientation (O).

2.5 Value Iteration

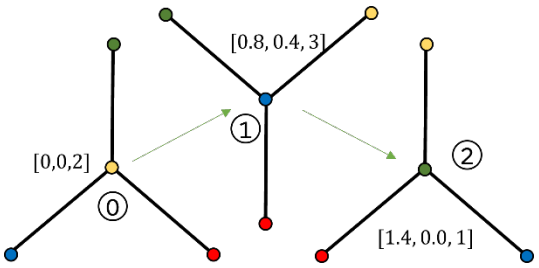
A standard finite horizon backward propagation value iteration algorithm was developed. The algorithm automatically rejects the actions produced out of the range states. The transition does not have a cost associated with it. Also, the discount factor is 1.

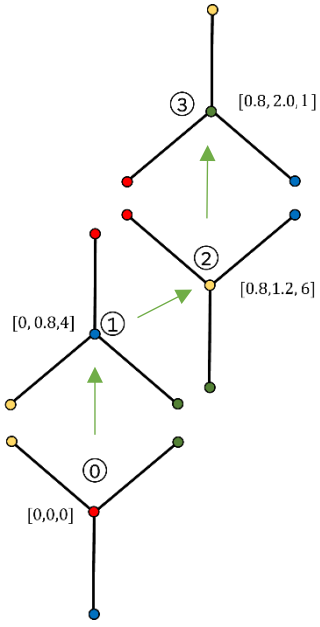
$$V_i(x) = \min_{u_{i-1}}(V_{i-1}(x'))$$
$$u_{K-i}(x) = argmax_u(V_{i-1}(x'))$$

Here, K is the horizon, and the Value function is initialized with $+\infty$.

3 Validation and Results

To validate the system, 2 goal locations are given to the Value Iteration algorithm. Initial training is for a 2-step horizon, and the second training is for a 3-step horizon. The initial state and the final state are depicted in the following image. The video evidence can be found in this link.





Due to the stochastic nature of the robot, the robot might not reach to the goal location. Hence, to ensure this, I also initialized the neighbor grids with zero cost.

4 Support materials

GitHub: https://github.com/DulanjanaPerera/ValueIterationSoftRobot.git

5 Reference

[1] C. Lee *et al.*, 'Soft robot review', *Int. J. Control Autom. Syst.*, vol. 15, no. 1, pp. 3–15, Feb. 2017, doi: 10.1007/s12555-016-0462-3.

[2] 'A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles | IEEE Journals & Magazine | IEEE Xplore'. Accessed: Oct. 30, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7490340?casa_token=YS9wnxa9ja0AAAAA:fD__hwTpRVG_5S4r3u9ypzlGpYfQ74pQRB3_BXzuKv-8s-dBFdhEN0jNfzKIWjmZH7QpNlnG

[3] D. Khalidi, D. Gujarathi, and I. Saha, 'A Heuristic Search Based Path Planning Algorithm for Temporal Logic Specifications', in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 8476–8482. doi: 10.1109/ICRA40945.2020.9196928.

[4] C. Wang, C. Cheng, D. Yang, G. Pan, and F. Zhang, 'Path Planning in Localization Uncertaining Environment Based on Dijkstra Method', *Frontiers in Neurorobotics*, vol. 16, 2022, Accessed: Oct. 28, 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnbot.2022.821991

[5] N. Yu, C. Wang, F. Mo, and J. Cai, 'Dynamic environment path planning based on Q-learning algorithm and genetic algorithm', *Journal of Beijing University of technology*, vol. 43, no. 7, pp. 1009–1016, 2017.

[6] C. Yan, X. Xiang, and C. Wang, 'Towards Real-Time Path Planning through Deep Reinforcement Learning for a UAV in Dynamic Environments', *J Intell Robot Syst*, vol. 98, no. 2, pp. 297–309, May 2020, doi: 10.1007/s10846-019-01073-3.

[7] 'Dynamic path planning of a three-dimensional underwater AUV based on an adaptive genetic algorithm - ScienceDirect'. Accessed: Oct. 28, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0029801822017073?casa_token=UVSS4SkJCzcAAAAA:ii07iegUPq6GFB4hFarE9yGgnmYa5OcpT_Yig0OUo0zCmK2iJZirIVgLh-C3LuO08AxbqAOgew

[8] R. Gonzalez, M. Kloetzer, and C. Mahulea, 'Comparative study of trajectories resulted from cell decomposition path planning approaches', in *2017 21st International Conference on System Theory, Control and Computing (ICSTCC)*, Oct. 2017, pp. 49–54. doi: 10.1109/ICSTCC.2017.8107010.

[9] I.-B. Jeong, S.-J. Lee, and J.-H. Kim, 'Quick-RRT*: Triangular inequality-based implementation of RRT* with improved initial solution and convergence rate', *Expert Systems with Applications*, vol. 123, pp. 82–90, Jun. 2019, doi: 10.1016/j.eswa.2019.01.032.

[10] M. Baumann, S. Léonard, E. A. Croft, and J. J. Little, 'Path Planning for Improved Visibility Using a Probabilistic Road Map', *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 195–200, Feb. 2010, doi: 10.1109/TRO.2009.2035745.

[11] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, 'A Survey of Autonomous Driving: Common Practices and Emerging Technologies', *IEEE Access*, vol. 8, pp. 58443–58469, 2020, doi: 10.1109/ACCESS.2020.2983149.

[12] H. Li, P. Li, L. Yang, J. Zou, and Q. Li, 'Safety research on stabilization of autonomous vehicles based on improved-LQR control', *AIP Advances*, vol. 12, no. 1, p. 015313, Jan. 2022, doi: 10.1063/5.0078950.

[13] C. Liu, S. Lee, S. Varnhagen, and H. E. Tseng, 'Path planning for autonomous vehicles using model predictive control', in *2017 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2017, pp. 174–179. doi: 10.1109/IVS.2017.7995716.

[14] D. M. Perera *et al.*, 'Teleoperation of Soft Modular Robots: Study on Real-time Stability and Gait Control', in *2023 IEEE International Conference on Soft Robotics (RoboSoft)*, Apr. 2023, pp. 01–07. doi: 10.1109/RoboSoft55895.2023.10122121.