Tổng quan về Lập kế hoạch đường đi

Ngày 6 tháng 4 năm 2025

1 Giới thiệu bài toán

Bài toán lập kế hoạch đường đi (path planning) là một bài toán tìm kiếm một chuỗi các cấu hình hợp lệ để di chuyển một đối tượng từ một trạng thái bắt đầu (source/start configuration S hoặc x_{init}) đến một trạng thái đích (destination/goal configuration G hoặc x_{goal} hoặc goal region $G(x_{goal})$).

Một cách cụ thể hơn, bài toán lập kế hoạch đường đi là tính toán một đường đi liên tục tối ưu kết nối trạng thái bắt đầu S và trạng thái đích G, đồng thời tránh va chạm với các chướng ngại vật đã biết trong không gian trạng thái tự do (X_{free}).

Bài toán lập kế hoạch tối ưu được định nghĩa là việc tìm kiếm đường đi tối thiểu hóa một hàm chi phí ($c(\sigma)$ hoặc $s(\sigma)$) trên đường đi đó, trong số tất cả các đường đi hợp lệ. Hàm chi phí thường là chiều dài của đường đi đã đi, nhưng cũng có thể là các tiêu chí khác như thời gian thực hiện hoặc năng lượng tiêu thụ.

Để giải quyết bài toán này, cần xem xét:

- Không gian trạng thái (X): Không gian chứa tất cả các cấu hình có thể có của robot.
- Không gian tự do (X_{free}): Phần của không gian trạng thái không chứa chướng ngại vật.
- Trạng thái bắt đầu (x_{init}): Cấu hình ban đầu của robot.
- Trạng thái đích (x_{goal} hoặc goal region $G(x_{goal})$): Cấu hình hoặc vùng cấu hình mà robot cần đạt tới.
- Đường đi (σ): Một chuỗi liên tục các trạng thái kết nối trạng thái bắt đầu và trạng thái đích. Một đường đi khả thi ($\sigma \in \mathcal{S}$ hoặc Σ) là một đường đi không có va chạm và thỏa mãn các điều kiện bắt đầu và kết thúc.

- Hàm chi phí ($c(\sigma)$ hoặc $s(\sigma)$): Một hàm ánh xạ mỗi đường đi khả thi đến một số thực dương, biểu thị "chi phí" của việc đi theo đường đi đó. Mục tiêu cuối cùng là tìm ra đường đi tối ưu (σ^*) trong tập hợp tất cả các đường đi khả thi, sao cho chi phí của nó là nhỏ nhất.

Bài toán path planning có thể được định nghĩa bởi bộ ba (X_{free}, x_{init}, X_{goal}). Trong bối cảnh các thuật toán như RRT và RRT*, bài toán này thường được giải quyết bằng cách khám phá không gian trạng thái để tìm ra một đường đi khả thi, và trong trường hợp của RRT*, cố gắng cải thiện đường đi đó để đạt được sự tối ưu. Neural RRT* tiếp cận bài toán bằng cách học một phân bố xác suất có hướng để tăng tốc quá trình tìm kiếm đường đi tối ưu.

2 Phân loại các thuật toán path planning

Các thuật toán path planning có thể được phân loại theo nhiều tiêu chí khác nhau. Dưới đây là một số cách phân loại chính:

Phân loại theo cách tiếp cận

- *Thuật toán dựa trên tìm kiếm (Search-based algorithms)*: Đây là các thuật toán truyền thống dựa trên việc tìm kiếm trên đồ thị được tạo ra từ bản đồ môi trường. Các thuật toán tiêu biểu bao gồm Dijkstra và A*. Các thuật toán này xử lý và tìm kiếm trên các đường đi được hình thành bởi các nút trên đồ thị để tìm ra đường đi tối ưu hoặc gần tối ưu. Giải pháp của bài toán sẽ luôn là duy nhất cho mỗi đầu vào cố định.

- *Thuật toán dựa trên lấy mẫu (Sampling-based path planning)*: Các thuật toán này xây dựng một cây hoặc đồ thị bằng cách lấy mẫu ngẫu nhiên trong không gian trạng thái và kết nối các mẫu này lại với nhau. Các thuật toán tiêu biểu bao gồm RRT (Rapidly-exploring Random Tree) và RRT*. *Phương pháp lấy mẫu* có thể chia thành phương pháp không thiên vị (lấy mẫu đồng nhất) trên toàn bộ không gian cấu hình và phương pháp thiên vị (biased method) hướng việc lấy mẫu đến các vùng quan trọng của không gian cấu hình.

- *Thuật toán dựa trên học (Learning-based path planning)*: Các thuật toán này sử dụng các mô hình học máy, chẳng hạn như mạng nơ-ron, để học cách lập kế hoạch đường đi hoặc cải thiện hiệu suất của các thuật toán truyền thống. Một ví dụ là Neural RRT*.

Phân loại theo mức độ hiểu biết về thông tin môi trường

- *Lập kế hoạch đường đi toàn cục (Global path planning)*: Thông tin về môi trường được biết đầy đủ.
- *Lập kế hoạch đường đi cục bộ (Local path planning)*: Thông tin về môi trường chỉ được biết một phần hoặc hoàn toàn không biết.

Phân loại theo tính chất của môi trường (tĩnh hoặc động)

- *Lập kế hoạch tĩnh (Static planning)*: Môi trường không thay đổi, các chướng ngại vật đứng yên.
- *Lập kế hoạch động (Dynamic planning)*: Các chướng ngại vật di chuyển trong môi trường. Các thuật toán cần phản ứng với những thay đổi này.

Phân loại theo tính chất của môi trường (tĩnh hoặc động)

Theo số lượng truy vấn:

- *Lập kế hoạch đơn truy vấn (Single-query)*: Tìm đường đi giữa một cặp điểm bắt đầu và kết thúc cụ thể (ví dụ: RRT, RRT*).
- *Lập kế hoạch đa truy vấn (Multi-query)*: Xây dựng một bản đồ đường đi có thể sử dụng cho nhiều truy vấn khác nhau (ví dụ: PRM).

Như vậy, có nhiều cách để phân loại các thuật toán path planning, tùy thuộc vào tiêu chí mà chúng ta quan tâm. Các thuật toán hiện đại thường kết hợp các ý tưởng từ các cách tiếp cận khác nhau để giải quyết các bài toán phức tạp trong thực tế. Ví dụ, Neural RRT* kết hợp phương pháp lấy mẫu của RRT* với khả năng học hỏi từ dữ liệu của mạng nơ-ron.

3 Nhóm thuật toán dựa trên lấy mẫu (Sampling-based path planning)

3.1 Thuật toán RRT (Rapidly-exploring Random Tree) và các biến thể

3.1.1 Thuật toán RRT (Rapidly-exploring Random Tree)

RRT là một thuật toán lập kế hoạch đường đi dựa trên việc lấy mẫu ngẫu nhiên và xây dựng một cây để khám phá không gian trạng thái. Thuật toán hoạt động theo các bước chính sau:

- **Khởi tạo**: Bắt đầu với một cây chỉ chứa trạng thái bắt đầu (x_{init}). Tập hợp các cạnh E ban đầu rỗng.
- **Lặp**: Thực hiện lặp lại n lần:
  - Lấy mẫu ngẫu nhiên: Lấy một mẫu ngẫu nhiên x_{rand} từ không gian tự do (SampleFreei). Quá trình lấy mẫu này thường là ngẫu nhiên đồng nhất trên không gian trạng thái, mặc dù các biến thể có thể sử dụng các phương pháp lấy mẫu khác.
  - Tìm nút gần nhất: Tìm nút $x_{nearest}$ trong cây hiện tại là nút gần nhất với x_{rand} (Nearest).
  - Tạo nút mới: Tạo một trạng thái mới x_{new} bằng cách di chuyển một bước từ $x_{nearest}$ theo hướng của x_{rand} (Steer). Hàm Steer giới hạn khoảng cách giữa $x_{nearest}$ và x_{new} để đảm bảo sự tăng trưởng của cây một cách có kiểm soát.
  - Kiểm tra va chạm: Kiểm tra xem đường đi giữa $x_{nearest}$ và x_{new} có bị va chạm với chướng ngại vật không (ObtacleFree).
  - Thêm vào cây: Nếu đường đi không có va chạm, thêm x_{new} vào tập hợp các đỉnh V của cây và thêm cạnh $(x_{nearest}, x_{new})$ vào tập hợp các cạnh E.
- Trả về: Sau khi lặp lại đủ số lần, thuật toán trả về cây $G = (V, E)$. Đường đi từ trạng thái bắt đầu đến một trạng thái trong tập mục tiêu (nếu đạt được) có thể được trích xuất từ cây này.

Đặc điểm của RRT:

- Khám phá hiệu quả: RRT hiệu quả trong việc khám phá các không gian trạng thái lớn và thậm chí cả không gian có số chiều cao.

- Phương pháp chiếu tiến: RRT là một phương pháp chiếu tiến, không yêu cầu giải quyết bài toán giá trị biên (kết nối trực tiếp điểm bắt đầu và điểm kết thúc bằng một bộ lập kế hoạch cục bộ).
- Tính đầy đủ xác suất: RRT được chứng minh là đầy đủ xác suất, nghĩa là nếu tồn tại một đường đi trong không gian tự do, RRT sẽ tìm thấy nó với xác suất tiến tới 1 khi số lượng mẫu tăng lên.

3.1.2 RRT star (RRT*)

RRT* là một biến thể của RRT được thiết kế để hội tụ đến nghiệm tối ưu khi số lượng mẫu tăng lên. Nó cải thiện RRT bằng cách thêm hai bước quan trọng:

- Chọn cha tốt nhất (Best Parent Selection): Khi một nút mới x_{new} được tạo, thay vì chỉ kết nối nó với nút gần nhất $x_{nearest}$, RRT* xem xét một tập hợp các nút lân cận (trong một bán kính nhất định) và chọn nút làm cha sao cho chi phí đường đi từ gốc đến x_{new} là nhỏ nhất. Điều này liên quan đến việc tính toán chi phí đường đi qua từng nút lân cận có thể làm cha và chọn nút mang lại chi phí thấp nhất mà không gây ra va chạm.
- Rewiring (Tái cấu trúc cây): Sau khi x_{new} được thêm vào cây, RRT* lại xem xét các nút lân cận của x_{new}. Đối với mỗi nút lân cận $x_{neighbor}$, thuật toán kiểm tra xem việc tái kết nối $x_{neighbor}$ với x_{new} làm cha có làm giảm chi phí đường đi từ gốc đến $x_{neighbor}$ hay không. Nếu có và đường đi mới không có va chạm, cây sẽ được tái cấu trúc bằng cách thay đổi cha của $x_{neighbor}$ thành x_{new}. Bước này cho phép cây "tối ưu hóa" dần dần các đường đi đã tìm thấy.

Đặc điểm của RRT*:

- Bán kính tìm kiếm lân cận (γ): RRT* sử dụng một bán kính tìm kiếm γ để xác định các nút lân cận cần xem xét cho việc chọn cha và rewiring. Giá trị của γ có vai trò quan trọng trong việc đảm bảo tính tối ưu tiệm cận của thuật toán. Theo, điều kiện để RRT* đạt được tính tối ưu tiệm cận là $\gamma > (2(1+1/d))^{1/d}(\frac{\mu(X_{free})}{\xi_d})^{1/d}$, trong đó d là số chiều của không gian trạng thái, $\mu(X_{free})$ là độ đo Lebesgue của không gian tự do, và ξ_d là thể tích của quả cầu đơn vị trong không gian d chiều.
- Tính tối ưu tiệm cận: RRT* được chứng minh là tối ưu tiệm cận, nghĩa là khi số lượng mẫu tăng lên vô hạn, chi phí của đường đi tốt nhất được

tìm thấy bởi RRT* sẽ hội tụ đến chi phí của đường đi tối ưu. RRT ban đầu không có tính chất này; nó có thể tìm thấy một đường đi khả thi nhưng không đảm bảo là đường đi ngắn nhất.

3.1.3 Neural RRT*

Neural RRT* là một thuật toán lập kế hoạch đường đi tối ưu mới dựa trên mạng nơ-ron tích chập (CNN) và thuật toán RRT*. NRRT* sử dụng một phân bố lấy mẫu không đồng nhất được tạo ra từ mô hình CNN để hướng dẫn quá trình lấy mẫu.

- Học phân bố xác suất: Mô hình CNN được huấn luyện bằng cách sử dụng một lượng lớn các trường hợp lập kế hoạch đường đi thành công, thường được tạo ra bằng thuật toán A* để có được các đường đi tối ưu trên bản đồ. Mô hình này học cách dự đoán phân bố xác suất của đường đi tối ưu trên bản đồ.

- Lấy mẫu có hướng: Phân bố xác suất dự đoán được sử dụng để làm lệch quá trình lấy mẫu trong RRT*, ưu tiên các vùng có khả năng cao chứa đường đi tối ưu. Điều này giúp NRRT* khám phá không gian trạng thái hiệu quả hơn và hội tụ nhanh hơn đến nghiệm tối ưu.

- Các kết quả mô phỏng cho thấy NRRT* có hiệu suất thuyết phục so với các thuật toán lập kế hoạch đường đi hiện đại, đạt được sự hội tụ nhanh hơn và đường đi ban đầu gần với đường đi tối ưu hơn. Tuy nhiên, hiệu suất của NRRT* phụ thuộc vào chất lượng của bộ dữ liệu huấn luyện. NRRT* cũng được coi là tối ưu tiệm cận theo các tính chất tương tự như RRT*.

Tóm lại, RRT là một thuật toán khám phá không gian hiệu quả, trong khi RRT* cải tiến bằng cách tối ưu hóa đường đi để tìm ra nghiệm tối ưu tiệm cận. Neural RRT* tiếp tục phát triển ý tưởng này bằng cách sử dụng học sâu để hướng dẫn quá trình khám phá, nhằm đạt được hiệu suất tốt hơn trong nhiều ứng dụng.

3.2 Thuật toán Probabilistic Roadmap (PRM)

Thuật toán Probabilistic Roadmap (PRM) là một thuật toán lập kế hoạch đường đi dựa trên lấy mẫu, đặc biệt phù hợp cho các bài toán lập kế hoạch đa truy vấn. Điều này có nghĩa là PRM xây dựng một đồ thị (roadmap) có thể được sử dụng để giải quyết các bài toán tìm đường đi giữa các điểm bắt đầu và kết thúc tùy ý trong không gian tự do.

- Ứng dụng cho đa truy vấn: PRM được thiết kế để giải quyết nhiều bài toán lập kế hoạch đường đi khác nhau trên cùng một bản đồ. Sau khi xây dựng xong roadmap, ta có thể nhanh chóng tìm kiếm đường đi giữa bất kỳ cặp điểm nào trên roadmap.

- Xây dựng Roadmap: Thuật toán PRM hoạt động bằng cách:

  – Khởi tạo tập đỉnh với điều kiện ban đầu.

  – Lấy mẫu ngẫu nhiên n điểm từ không gian tự do (X_{free}).

  – Cố gắng kết nối các điểm trong phạm vi khoảng cách r. Quy tắc kết nối tương tự như PRM, nhưng cho phép kết nối giữa các đỉnh trong cùng một thành phần liên thông (điểm này là sự khác biệt so với phiên bản "đơn giản hóa" được gọi là sPRM).

- Tính tối ưu:

  – Thuật toán PRM không tối ưu tiệm cận. Sự thiếu tối ưu này là do cách xây dựng tăng dần và ràng buộc loại bỏ các cạnh tạo ra các kết nối không cần thiết bên trong một thành phần liên thông.

  – Ngược lại, thuật toán sPRM (simplified PRM), với cách xây dựng theo lô (batch construction) và không có ràng buộc trên, thực sự là tối ưu tiệm cận. Theo nguồn, đồ thị được trả về bởi sPRM bao gồm tất cả các đường đi có trong đồ thị được trả về bởi PRM*. Tính tối ưu tiệm cận của sPRM theo sau tính tối ưu tiệm cận của PRM*.

- Quan hệ với RRT: Nguồn chỉ ra rằng, với một điểm bắt đầu cụ thể (được lấy mẫu từ không gian tự do), xác suất tìm thấy đường đi của PRM tương đương với RRT cho cùng số lượng mẫu. Tuy nhiên, PRM không đảm bảo tính tối ưu tiệm cận như RRT*.

- Heuristics trong thực tế: Nguồn cũng lưu ý rằng các heuristics thường được sử dụng trong việc triển khai PRM và sPRM trên thực tế có thể không tối ưu tiệm cận. Ví dụ, thuật toán k-nearest sPRM (sPRM kết nối mỗi nút với k nút lân cận nhất) không tối ưu tiệm cận.

Tóm lại, thuật toán PRM là một phương pháp hiệu quả cho bài toán lập kế hoạch đường đi đa truy vấn bằng cách xây dựng một đồ thị ngẫu nhiên. Mặc dù phiên bản cơ bản của PRM không tối ưu tiệm cận, phiên bản đơn giản hóa sPRM (với sự khác biệt chính trong cách kết nối) lại có tính chất này, dù có thể phức tạp hơn về mặt tính toán.

3.3 Nhóm thuật toán dựa trên tìm kiếm trên đồ thị (Search-based algorithms)

Thuật toán dựa trên tìm kiếm (Search-based algorithms) tiêu biểu gồm Dijkstra và A*.

3.3.1 Thuật toán Dijkstra

- Đây là một thuật toán tìm đường đi ngắn nhất từ một nút nguồn đến tất cả các nút khác trong một đồ thị [lịch sử trò chuyện].
- Thuật toán này hoạt động trên đồ thị có các cạnh được gán trọng số không âm [lịch sử trò chuyện].
- Dijkstra duy trì một tập hợp các nút đã biết đường đi ngắn nhất từ nguồn và một tập hợp các nút chưa biết.
- Ban đầu, chỉ có nút nguồn được biết và khoảng cách đến chính nó là 0. Khoảng cách đến tất cả các nút khác là vô cùng.
- Trong mỗi bước, thuật toán chọn nút chưa biết có khoảng cách từ nguồn nhỏ nhất, đánh dấu nó là đã biết, và cập nhật khoảng cách đến các nút lân cận của nó nếu đi qua nút vừa chọn sẽ tạo ra đường đi ngắn hơn.
- Quá trình này tiếp tục cho đến khi tất cả các nút đã được đánh dấu là đã biết hoặc nút đích đã được tìm thấy.
- Thuật toán Dijkstra đảm bảo tìm được đường đi ngắn nhất từ nút nguồn đến bất kỳ nút nào khác trong đồ thị nếu tồn tại đường đi [lịch sử trò chuyện].

3.3.2 Thuật toán A* (A-star)

- Đây là một thuật toán tìm đường đi ngắn nhất có thông tin (informed search algorithm).
- Tương tự như Dijkstra, A* cũng hoạt động trên một đồ thị.
- Điểm khác biệt chính của A* là nó sử dụng một hàm heuristic để ước tính chi phí từ nút hiện tại đến nút mục tiêu.
- Hàm đánh giá $f(n)$ cho mỗi nút n trong A* thường được tính như sau: $f(n) = g(n) + h(n)$, trong đó:

  - $g(n)$ là chi phí thực tế đã đi từ nút bắt đầu đến nút n.
  - $h(n)$ là chi phí ước tính (heuristic) từ nút n đến nút mục tiêu.

- A* duy trì một hàng đợi ưu tiên các nút cần khám phá, được ưu tiên dựa trên giá trị $f(n)$ thấp nhất.

- Trong mỗi bước, thuật toán chọn nút có $f(n)$ thấp nhất từ hàng đợi, đánh dấu nó là đã được khám phá, và thêm các nút lân cận của nó vào hàng đợi (nếu cần).

- A* thường đảm bảo tìm được đường đi tối ưu nếu hàm heuristic là chấp nhận được (admissible), nghĩa là nó không bao giờ đánh giá quá cao chi phí thực tế để đến mục tiêu [lịch sử trò chuyện].

- Do sử dụng thông tin heuristic, A* thường khám phá không gian tìm kiếm một cách hiệu quả hơn so với các thuật toán không có thông tin như Dijkstra, đặc biệt trong các không gian tìm kiếm lớn.

Cả Dijkstra và A* đều là các thuật toán dựa trên tìm kiếm vì chúng hoạt động bằng cách tìm kiếm trên một đồ thị biểu diễn không gian môi trường. Chúng đều đảm bảo tìm được đường đi tối ưu nếu nó tồn tại. thị

3.4 Batch-Informed RRT* (BIT*)

Thuật toán BIT* là một thuật toán lập kế hoạch đường đi dựa trên việc kết hợp các ưu điểm của cả phương pháp tìm kiếm trên đồ thị và phương pháp dựa trên lấy mẫu. BIT* hoạt động bằng cách nhận ra rằng một tập hợp các mẫu mô tả một đồ thị hình học ngẫu nhiên (RGG) ẩn. Điều này cho phép BIT* kết hợp hiệu quả tìm kiếm có thứ tự của các kỹ thuật dựa trên đồ thị, như A*, với khả năng mở rộng theo thời gian thực của các thuật toán dựa trên lấy mẫu, như RRT.

- Ý tưởng chính của thuật toán

  - Các thuật toán dựa trên lấy mẫu như RRT* có tính tối ưu bất tiệm cận nhưng có thể chậm trong việc tìm ra nghiệm tốt, đặc biệt là trong không gian trạng thái phức tạp hoặc chiều cao.
  - Các phương pháp tìm kiếm trên đồ thị như A* hiệu quả khi không gian rời rạc hoặc được rời rạc hóa tốt, nhưng khó áp dụng trực tiếp cho không gian liên tục.

  - BIT* tận dụng lợi thế của việc xử lý các mẫu theo lô (batch) để thực hiện tìm kiếm có thứ tự trên không gian lập kế hoạch liên tục, đồng thời duy trì hiệu suất theo thời gian thực.
  - BIT* sử dụng hàm heuristic để hướng dẫn hiệu quả việc tìm kiếm trong một loạt các RGG ẩn ngày càng dày đặc, đồng thời tái sử dụng thông tin từ các lần tìm kiếm trước.

- Tóm tắt thuật toán

  1. Khởi tạo:
     * Tập hợp đỉnh V ban đầu chứa trạng thái bắt đầu (xstart).
     * Tập hợp cạnh E ban đầu rỗng.
     * Tập hợp mẫu Xsamples ban đầu chứa trạng thái mục tiêu (xgoal).
     * Hai hàng đợi ưu tiên được sử dụng: QE (hàng đợi cạnh) và QV (hàng đợi đỉnh), ban đầu rỗng.
     * Bán kính r của RGG ẩn ban đầu được đặt thành vô cùng.
  2. Lặp lại theo lô (Batch): Thuật toán lặp lại cho đến khi tìm thấy nghiệm hoặc đạt đến điều kiện dừng. Mỗi lần lặp bắt đầu một lô mới khi cả QE và QV đều rỗng.
  3. Tỉa cây và mẫu: Các trạng thái và cây bao trùm được tỉa bỏ nếu chúng không thể cải thiện nghiệm hiện tại.
  4. Tạo lô mẫu mới: Một tập hợp gồm m mẫu mới được thêm vào RGG từ vùng không gian con có khả năng chứa nghiệm tốt hơn. Việc lấy mẫu này có thể được thực hiện bằng phương pháp loại bỏ dựa trên heuristic hoặc lấy mẫu trực tiếp (ví dụ: sử dụng phương pháp lấy mẫu có thông tin như trong Informed RRT*).
  5. Gán nhãn và đưa vào hàng đợi: Các đỉnh trong cây được gán nhãn để chỉ xem xét các kết nối đến các trạng thái mới. Các đỉnh này sau đó được đưa trở lại hàng đợi QV để mở rộng.
  6. Cập nhật bán kính RGG: Bán kính r của RGG ẩn được cập nhật để phản ánh kích thước hiện tại của nó (q - tổng số mẫu).
  7. Tìm kiếm Heuristic: Một cây rõ ràng được xây dựng từ trạng thái bắt đầu hướng tới mục tiêu bằng cách sử dụng tìm kiếm dựa trên heuristic. Quá trình này sử dụng hàng đợi cạnh QE, nơi các cạnh được sắp xếp theo ước tính chi phí giải pháp tối thiểu đi qua cạnh đó (kết hợp chi phí đã đi và chi phí ước tính đến mục tiêu bằng heuristic).

8. Kết thúc lô: Việc tìm kiếm trong một lô kết thúc khi tìm thấy một nghiệm hoặc khi không thể mở rộng thêm cây. Nếu tìm thấy nghiệm, lô sẽ kết thúc và có thể bắt đầu một lô mới sau khi tỉa và thêm mẫu mới để tìm kiếm nghiệm tốt hơn.

- Đặc điểm của BIT*
  - Tối ưu bất tiệm cận: BIT* được chứng minh là tối ưu bất tiệm cận, nghĩa là nó sẽ hội tụ đến nghiệm tối ưu khi số lượng mẫu tăng lên vô hạn.
  - Hoàn chỉnh xác suất: BIT* cũng là hoàn chỉnh xác suất, nghĩa là nếu có một nghiệm tồn tại, BIT* sẽ tìm thấy nó với xác suất tiến đến 1 khi số lượng mẫu tăng lên vô hạn.
  - Hiệu suất theo thời gian thực (Anytime Performance): BIT* có khả năng tìm ra các nghiệm khả thi nhanh chóng và tiếp tục cải thiện chất lượng nghiệm theo thời gian.
  - Tìm kiếm có thông tin: Việc sử dụng hàm heuristic và tập trung lấy mẫu vào vùng có khả năng chứa nghiệm tốt hơn giúp BIT* tìm kiếm hiệu quả hơn so với RRT*.
  - Hiệu quả trong không gian chiều cao: Các thử nghiệm cho thấy BIT* hoạt động tốt hơn các thuật toán khác, bao gồm RRT*, Informed RRT* và FMT*, đặc biệt là trong không gian trạng thái có chiều cao.
  - Tái sử dụng thông tin: BIT* tái sử dụng thông tin từ các lô tìm kiếm trước, giúp tăng tốc độ hội tụ.
  - Cân bằng giữa thăm dò và khai thác: Bằng cách sử dụng tìm kiếm heuristic theo lô và lấy mẫu có thông tin, BIT* cân bằng giữa việc thăm dò không gian trạng thái và khai thác các vùng hứa hẹn.

- Mối quan hệ với các thuật toán khác:
  - RRT*: BIT* có thể được xem như một sự tổng quát hóa của RRT*.
  - Informed RRT*: Với kích thước lô là một mẫu (m=1), BIT* tương tự như Informed RRT*, tập trung tìm kiếm trong vùng elip có thông tin.
  - Fast Marching Trees: Với một lô duy nhất và heuristic bằng không, BIT* tương tự như FMT*.

  - Lifelong Planning A (LPA*): Hàng đợi cạnh QE của BIT* là một mở rộng của hàng đợi đỉnh của LPA* cho các bài toán trong không gian liên tục, bao gồm cả ước tính heuristic về chi phí cạnh.
  - Hàm heuristic: Được sử dụng để ước tính chi phí từ một trạng thái đến mục tiêu.
  - Hằng số RGG (r hoặc k): Xác định cách các cạnh được hình thành trong RGG ẩn.
  - Số lượng mẫu trên mỗi lô (m): Xác định số lượng mẫu mới được thêm vào trong mỗi lần lặp.

Tóm lại, BIT* là một thuật toán lập kế hoạch đường đi mạnh mẽ, kết hợp các ưu điểm của tìm kiếm đồ thị và lấy mẫu, mang lại hiệu suất tốt trong nhiều loại bài toán, đặc biệt là những bài toán phức tạp và có chiều cao.

4 Nhóm Thuật toán dựa trên học (Learning-based path planning)

Nhóm Thuật toán dựa trên học (Learning-based path planning) sử dụng các mô hình học máy để học cách lập kế hoạch đường đi hoặc cải thiện hiệu suất của các thuật toán truyền thống. Các thuật toán này có tiềm năng hội tụ nhanh hơn và tìm ra các đường đi chất lượng cao hơn bằng cách học từ dữ liệu. Trong các nguồn tài liệu, hai thuật toán tiêu biểu thuộc nhóm này được đề cập chi tiết là Neural RRT* và Neural Informed RRT*.

4.1 Neural RRT*

RRT và các biến thể của nó có khả năng khám phá không gian trạng thái nhanh chóng và hiệu quả. Tuy nhiên, chúng có thể nhạy cảm với giải pháp ban đầu và hội tụ chậm đến giải pháp tối ưu, dẫn đến việc tiêu tốn nhiều bộ nhớ và thời gian.

- Ý tưởng chính: Neural RRT* đề xuất một mô hình mạng nơ-ron tích chập (CNN) để dự đoán phân phối xác suất của đường đi tối ưu trên bản đồ. Phân phối xác suất này sau đó được sử dụng để hướng dẫn quá trình lấy mẫu, tập trung vào các vùng có khả năng chứa đường đi tối ưu.
- Mô hình CNN:

  - Đầu vào: Mô hình CNN nhận đầu vào là ảnh 2D biểu diễn không gian trạng thái (0: tự do, 1: chướng ngại vật, 2: điểm bắt đầu, 3: điểm kết thúc), kích thước bước tối đa của robot và khoảng cách an toàn tối thiểu từ đường đi đến chướng ngại vật.
  - Đầu ra: Đầu ra của CNN là một bản đồ xác suất 2D, trong đó mỗi pixel thể hiện xác suất điểm đó nằm trên đường đi tối ưu.

- Huấn luyện: Mô hình được huấn luyện bằng cách sử dụng thuật toán A* để tạo ra các đường đi tối ưu làm dữ liệu ground truth. Hàm mất mát cross-entropy được sử dụng để so sánh bản đồ xác suất dự đoán với ground truth. Dữ liệu huấn luyện bao gồm nhiều bản đồ ngẫu nhiên 2D với các cài đặt tham số khác nhau (khoảng cách an toàn và kích thước bước) và các cặp điểm bắt đầu - kết thúc ngẫu nhiên.

- Quá trình lập kế hoạch: Neural RRT* kết hợp việc lấy mẫu từ phân phối xác suất dự đoán (lấy mẫu không đồng đều) và lấy mẫu đồng đều để duy trì khả năng khám phá toàn bộ không gian trạng thái. Thuật toán sử dụng một tỷ lệ α để kiểm soát tần suất lấy mẫu đồng đều và không đồng đều. Phần còn lại của quy trình tương tự như RRT*, bao gồm mở rộng cây, kết nối lại các nút và kiểm tra xem nút mới có nằm trong vùng mục tiêu hay không.

- Ưu điểm: Neural RRT* cho thấy khả năng hội tụ nhanh hơn, đường đi ban đầu gần với đường đi tối ưu hơn và ít tốn tài nguyên tính toán hơn so với RRT* và Informed RRT* trong các thử nghiệm. Nó cũng thể hiện tính ổn định và độ bền tốt khi các tham số như khoảng cách an toàn thay đổi.

- Nhược điểm: Thuật toán dựa vào chất lượng của bộ dữ liệu huấn luyện. Nếu phân phối xác suất dự đoán không chính xác, hiệu suất có thể bị ảnh hưởng.

4.2 Neural Informed RRT*

IRRT* dựa vào đường đi ban đầu, và chất lượng của đường đi này không được đảm bảo, đồng thời mất nhiều thời gian để hội tụ đến đường đi tối ưu. Các phương pháp dựa trên học khác mã hóa toàn bộ không gian trạng thái mà không có sự cải thiện lặp đi lặp lại và không xem xét tính kết nối của tập hợp trạng thái dẫn đường, ảnh hưởng đến tốc độ hội tụ trong các bài toán phức tạp.

- Ý tưởng chính: Neural Informed RRT* (NIRRT*) là một sự tăng cường của RRT* bằng cách sử dụng một mạng dựa trên điểm (PointNet++) để trực tiếp nhận các trạng thái tự do dưới dạng đám mây điểm làm đầu vào và tạo ra nhiều trạng thái dẫn đường trong một lần chạy. Nó tích hợp mạng này với Informed RRT* thông qua một cơ chế gọi là Neural Focus và đề xuất Neural Connect để giải quyết vấn đề kết nối của tập hợp trạng thái dẫn đường suy ra.

- PointNetGuide: Sử dụng PointNet++ làm mạng dựa trên điểm. Nó nhận đầu vào là trạng thái bắt đầu, trạng thái mục tiêu, độ dài đường đi tốt nhất hiện tại và các trạng thái tự do, sau đó xuất ra một tập hợp các trạng thái dẫn đường (X_{guide}). Đây là các trạng thái có xác suất cao hơn được chọn khi lấy mẫu.

- Huấn luyện PointNetGuide: Mạng được huấn luyện trên một bộ dữ liệu gồm nhiều thế giới ngẫu nhiên 2D. Thuật toán A* được sử dụng để tạo ra đường đi tối ưu theo từng pixel. Một đám mây điểm được tạo ra. Nhãn trạng thái dẫn đường được tạo bằng cách kiểm tra xem mỗi điểm có nằm gần bất kỳ điểm nào trên đường đi tối ưu (pixel-wise) trong bán kính η hay không.

- Neural Focus: Hạn chế đám mây điểm đầu vào của mạng dựa trên điểm nằm trong vùng tập trung (X_{focus}).

- Neural Connect: Sau khi tìm kiếm theo chiều rộng (BFS) hoàn tất, nếu tính kết nối vẫn chưa được xác nhận, NIRRT* tìm các điểm biên (X_{bound}) gần các điểm dẫn đường đã được thăm. Một điểm từ X_{bound} được chọn dựa trên chi phí heuristic để đạt được mục tiêu từ điểm bắt đầu và điểm được chọn, sau đó vai trò của điểm bắt đầu và điểm kết thúc được đảo ngược và quá trình BFS được lặp lại. PointNet++ sau đó được sử dụng để suy ra các trạng thái dẫn đường mới với điểm bắt đầu và kết thúc mới. Quá trình này tiếp tục cho đến khi kết nối được xây dựng hoặc đạt đến giới hạn số lần lặp.

- Ưu điểm: NIRRT* có thể tạo ra nhiều trạng thái dẫn đường trong một lần chạy và cố gắng giải quyết vấn đề kết nối của các trạng thái này.

- Nhược điểm: Các thử nghiệm cho thấy NIRRT* và IRRT* không khác biệt nhiều về việc tối ưu hóa chi phí đường đi, và IRRT* thậm chí còn nhanh hơn. NIRRT* vẫn có thể gặp khó khăn trong các hành lang hẹp do sự phụ thuộc vào giải pháp ban đầu.

Tóm lại, nhóm Thuật toán dựa trên học đang phát triển mạnh mẽ, kết hợp sức mạnh của học máy với các phương pháp lập kế hoạch đường đi truyền thống để đạt được hiệu suất tốt hơn, đặc biệt trong các môi trường phức tạp và các bài toán có độ khó cao. Neural RRT* tập trung vào việc học phân phối lấy mẫu tối ưu, trong khi Neural Informed RRT* cố gắng cải thiện RRT* bằng cách học cách tạo ra các trạng thái dẫn đường hiệu quả hơn.

5 Bài tập Nhóm

Mỗi nhóm chọn 1 - 2 thuật toán sau để trình bày chi tiết thuật toán, code và demo.

Yêu cầu trình bày: Viết báo cáo bài tập nhóm gồm các nội dung sauInformed (khuyến khích soạn thảo bằng Latex để thể hiện rõ các công thức):
- Giới thiệu thuật toán
- Trình bày thuật toán (chi tiết và có hình ảnh minh hoạ)thức.
- Code thuật toán (C++, python, Julia).
- Demo thuật toán.
- Nhận xét.

*__Chú ý__: Được tham khảo tất cả các nguồn các em tìm được nhưng phải tự đánh lại, tuyệt đối không cắt ghép công thức và thuật toán.

Các thuật toán bao gồm:

1. A* Algorithm
2. RRT
3. RRT*
4. Batch-Informed RRT* (BIT*)
5. Rapidly-exploring Random Graph (RRG)
6. Probabilistic Roadmap Method (PRM)
7. Neural RRT*
8. Neural Informed RRT*

Những thuật toán đánh với font đỏ khó hơn, nên sẽ bonus điểm nếu dũng cảm làm những thuật toán này :).

DEADLINE: 23h59, 26/4/2025. Mỗi lớp (Nhóm 11, 12, 13, 14) nộp theo các link dưới đây. Mỗi nhóm của mỗi lớp sẽ vào link của lớp mình, tạo folder và nộp bài vào folder đó. Folder ghi theo tên lớp sau đến tên nhóm, ví dụ: Nhóm 11.1 tức là nhóm số 1 của Nhóm lớp 11.

Link nộp bài Nhóm 11:
`https://drive.google.com/drive/folders/19rHcayaOdSF7KUTBwHAPjT3sM_2np2Xk?usp=sharing`

Link nộp bài Nhóm 12:
`https://drive.google.com/drive/folders/1EbUA4SCJBKvm8i43woryUcTWMr2GCcD8?usp=sharing`

Link nộp bài Nhóm 13:
`https://drive.google.com/drive/folders/1s1o1xo4BukggNofmuyApfEcR31AIsCjU?usp=sharing`

Link nộp bài Nhóm 14:
`https://drive.google.com/drive/folders/1_pFRh-T_ICddX-UbDBbYOSwenHm96Dc5?usp=sharing`

Tài liệu

[1] Tạ Trung Tín and Phạm Ngọc Long, *Project Report: Deep learning and sampling-based path planning for autonomous robots*, January 2025. Lecturer: Lê Hồng Trang.

[2] Steven M. LaValle, *Rapidly-exploring random trees: Progress and prospects*, Unpublished manuscript, University of Illinois, 1998. `http://msl.cs.uiuc.edu/~lavalle/papers/Lav98c.pdf`

[3] *Sampling-based Algorithms for Optimal Motion Planning*, `https://arxiv.org/pdf/1405.5848`

[4] Pieter Abbeel, *RRT**, `https://people.eecs.berkeley.edu/~pabbeel/cs287-fa19/optreadings/rrtstar.pdf`

[5] Jiankun Wang, Chenming Li, Chaoqun Wang, Max Q.-H. Meng, and Wenzheng Chi, *Neural-RRT*: Learning Near-Optimal Paths for Robot*

Motion Planning in Complex Environments, IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 1888-1895, April 2020.