HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN I

—————————o0o—————————

BÁO CÁO BÀI TẬP LỚN

Đề tài: **"THUẬT TOÁN PRM (PROBABILISTIC ROADMAP) VÀ ỨNG DỤNG TRONG LẬP TRÌNH DI CHUYỂN ROBOT"**

Giảng Viên Hướng Dẫn:	**TS. Nguyễn Kiều Linh**
Sinh viên thực hiện:	**Nguyễn Trường Thái**
Mã sinh viên:	**BXXDCCNYYY**
Lớp:	**DXHTTTY**
Niên khóa:	**20xx-20xx**
Hệ đào tạo:	**Đại học chính quy**

Hà Nội, 4/2025

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN I

—————————o0o—————————

BÁO CÁO BÀI TẬP LỚN

Đề tài: "THUẬT TOÁN PRM (PROBABILISTIC ROADMAP) VÀ ỨNG DỤNG TRONG LẬP TRÌNH DI CHUYỂN ROBOT"

Giảng Viên Hướng Dẫn:	TS. Nguyễn Kiều Linh
Sinh viên thực hiện:	Nguyễn Trường Thái
Mã sinh viên:	BXXDCCNYYY
Lớp:	DXHTTTY
Niên khóa:	20xx-20xx
Hệ đào tạo:	Đại học chính quy

Hà Nội, 4/2025

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Điểm: **(Bằng chữ:**)

Hà Nội, ngày tháng năm 20...

Giảng viên hướng dẫn

NHẬN XÉT CỦA GIẢNG VIÊN PHẢN BIỆN

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Điểm: **(Bằng chữ:)**

Hà Nội, ngày tháng năm 20...

Giảng viên phản biện

Mục lục

LỜI CẢM ƠN

Trước hết, tôi xin gửi lời cảm ơn chân thành và sâu sắc đến TS. Nguyễn Kiều Linh, người đã tận tình hướng dẫn, chỉ bảo và hỗ trợ tôi trong suốt quá trình thực hiện đồ án này. Những kiến thức, kinh nghiệm và sự nhiệt tình của cô đã giúp tôi hoàn thành đồ án một cách tốt nhất.

Tôi cũng xin gửi lời cảm ơn đến các thầy cô giáo trong Khoa Công nghệ Thông tin I, Học viện Công nghệ Bưu chính Viễn thông, những người đã truyền đạt kiến thức, kỹ năng và tạo điều kiện thuận lợi cho tôi trong suốt quá trình học tập và nghiên cứu.

Cuối cùng, tôi xin cảm ơn gia đình, bạn bè đã luôn bên cạnh, động viên và hỗ trợ tôi trong suốt thời gian qua.

Hà Nội, tháng 4 năm 2025

Sinh viên

Nguyễn Trường Thái

Danh sách hình vẽ

Danh sách bảng

Danh sách các ký hiệu và chữ viết tắt

PRM	Probabilistic Roadmap (Bản đồ xác suất)
RRT	Rapidly-exploring Random Tree (Cây ngẫu nhiên khám phá nhanh)
KNN	K-Nearest Neighbors (K láng giềng gần nhất)
UAV	Unmanned Aerial Vehicle (Phương tiện bay không người lái)
KDTree	K-Dimensional Tree (Cấu trúc dữ liệu cây K chiều)

Mở đầu

Trong lĩnh vực robotics, một trong những thách thức cơ bản là làm thế nào để robot có thể di chuyển an toàn và hiệu quả trong môi trường có chướng ngại vật. Bài toán này, được gọi là bài toán lập kế hoạch đường đi (path planning), đóng vai trò quan trọng trong nhiều ứng dụng thực tế như robot công nghiệp, xe tự hành, thiết bị bay không người lái (UAV), và robot phẫu thuật.

Trong số nhiều phương pháp giải quyết bài toán lập kế hoạch đường đi, thuật toán PRM (Probabilistic Roadmap) nổi bật như một giải pháp hiệu quả, đặc biệt trong môi trường phức tạp với nhiều chướng ngại vật. Thuật toán này dựa trên nguyên lý xác suất, tạo ra một "bản đồ đường đi" bằng cách sinh ngẫu nhiên các điểm trong không gian tự do và kết nối chúng lại với nhau, sau đó sử dụng các thuật toán tìm đường như Dijkstra hoặc A* để tìm đường đi tối ưu từ điểm xuất phát đến điểm đích.

Đồ án này tập trung nghiên cứu về thuật toán PRM, từ cơ sở lý thuyết đến triển khai thực tế trong lập trình di chuyển robot. Chúng tôi sẽ phân tích chi tiết các giai đoạn của thuật toán, cài đặt thuật toán bằng ngôn ngữ Python, và minh họa kết quả thông qua các hình ảnh trực quan. Ngoài ra, đồ án cũng so sánh PRM với các thuật toán lập kế hoạch đường đi khác như RRT (Rapidly-exploring Random Tree) và RRT*, đồng thời thảo luận về các ứng dụng thực tế của PRM trong nhiều lĩnh vực.

Mục tiêu của đồ án là cung cấp một cái nhìn toàn diện về thuật toán PRM, giúp người đọc hiểu rõ cách thức hoạt động, ưu nhược điểm, và tiềm năng ứng dụng của thuật toán này trong việc giải quyết bài toán lập kế hoạch đường đi cho robot.

Chương 1

Tổng quan về thuật toán PRM

1.1 Giới thiệu về bài toán lập kế hoạch đường đi

Lập kế hoạch đường đi (path planning) là một trong những bài toán cơ bản và quan trọng trong lĩnh vực robotics. Bài toán này đặt ra yêu cầu tìm một đường đi từ điểm xuất phát đến điểm đích, tránh các chướng ngại vật trong môi trường, và tối ưu hóa theo một số tiêu chí như khoảng cách, thời gian, hoặc năng lượng tiêu thụ.

Bài toán lập kế hoạch đường đi có nhiều ứng dụng thực tế trong các lĩnh vực như:

- Robot công nghiệp trong nhà máy
- Xe tự hành trên đường phố
- Thiết bị bay không người lái (UAV)
- Robot phẫu thuật trong y tế
- Robot dịch vụ trong khách sạn, nhà hàng

Có nhiều phương pháp để giải quyết bài toán lập kế hoạch đường đi, từ các phương pháp cổ điển như thuật toán A*, Dijkstra đến các phương pháp hiện đại dựa trên xác suất như PRM, RRT. Mỗi phương pháp đều có những ưu điểm và hạn chế riêng, phù hợp với các tình huống và yêu cầu khác nhau.

1.2 Khái niệm và nguyên lý của thuật toán PRM

Thuật toán PRM (Probabilistic Roadmap) là một phương pháp lập kế hoạch đường đi dựa trên xác suất, được giới thiệu lần đầu bởi Kavraki và cộng sự vào năm 1996. Thuật toán này hoạt động dựa trên nguyên lý tạo ra một "bản đồ đường đi" (roadmap) bằng cách sinh ngẫu nhiên các điểm trong không gian tự do và kết nối chúng lại với nhau.

Nguyên lý cơ bản của thuật toán PRM bao gồm hai giai đoạn chính:

1. **Giai đoạn xây dựng bản đồ (Learning Phase):** Trong giai đoạn này, thuật toán sinh ngẫu nhiên các điểm trong không gian tự do (không có chướng ngại vật), sau đó kết nối các điểm gần nhau bằng các cạnh nếu đường nối giữa chúng không đi qua chướng ngại vật. Kết quả là một đồ thị với các đỉnh là các điểm mẫu và các cạnh là các đường nối khả thi.

2. **Giai đoạn truy vấn (Query Phase):** Khi có yêu cầu tìm đường từ điểm xuất phát đến điểm đích, thuật toán sẽ kết nối hai điểm này vào bản đồ đã xây dựng, sau đó sử dụng các thuật toán tìm đường trên đồ thị như Dijkstra hoặc A* để tìm đường đi tối ưu.

Thuật toán PRM có một số ưu điểm nổi bật:

- Hiệu quả trong không gian có nhiều chiều
- Khả năng xử lý môi trường phức tạp với nhiều chướng ngại vật
- Giai đoạn xây dựng bản đồ chỉ cần thực hiện một lần, sau đó có thể sử dụng lại cho nhiều truy vấn khác nhau
- Dễ dàng mở rộng và tùy chỉnh cho các ứng dụng cụ thể

1.3 So sánh PRM với các thuật toán lập kế hoạch đường đi khác

Để hiểu rõ hơn về vị trí và vai trò của thuật toán PRM trong lĩnh vực lập kế hoạch đường đi, chúng ta cần so sánh nó với các thuật toán khác, đặc biệt là

RRT (Rapidly-exploring Random Tree) và RRT*.

Thuật toán	**Cách hoạt động**	**Ưu điểm**	**Nhược điểm**
PRM	Tạo đồ thị toàn cục trước rồi tìm đường	Nhanh với không gian tĩnh, nhiều lần dùng	Không hợp với robot động học phức tạp
RRT	Tạo cây từ start, mở rộng dần	Dễ dùng với không gian động, đơn giản	Đường đi xấu, không tối ưu
RRT*	Giống RRT, nhưng tối ưu hóa cây	Đường đi tốt hơn, gần tối ưu	Chậm hơn, phức tạp hơn

Bảng 1.1: So sánh các thuật toán lập kế hoạch đường đi

PRM vs RRT:

- PRM xây dựng một đồ thị toàn cục, trong khi RRT xây dựng một cây từ điểm xuất phát.
- PRM phù hợp với môi trường tĩnh và nhiều truy vấn, trong khi RRT phù hợp với môi trường động và truy vấn đơn.
- PRM yêu cầu nhiều tính toán trong giai đoạn xây dựng bản đồ, nhưng nhanh trong giai đoạn truy vấn, trong khi RRT phân bố tính toán đều hơn.

PRM vs RRT*:

- RRT* là phiên bản cải tiến của RRT, tập trung vào việc tối ưu hóa đường đi.
- PRM thường cho kết quả tối ưu hơn RRT, nhưng có thể không tối ưu bằng RRT* khi số lượng điểm mẫu tương đương.
- RRT* có khả năng hội tụ đến đường đi tối ưu khi số lượng điểm mẫu tăng lên, trong khi PRM không đảm bảo điều này.

Yêu cầu và hạn chế của PRM:

- PRM đòi hỏi map tĩnh do các nguyên nhân sau:

  - Đồ thị được xây dựng trước (offline) dựa trên vị trí chướng ngại vật cố định
  - Mọi thay đổi về vị trí vật cản đòi hỏi xây dựng lại toàn bộ roadmap
  - Thuật toán không có cơ chế cập nhật động các kết nối đã tồn tại

- Hạn chế chính:
  - Hiệu suất giảm mạnh trong không gian nhiều chiều
  - Khó xử lý các chướng ngại vật di động hoặc thay đổi hình dạng
  - Phụ thuộc nhiều vào chất lượng phân bố điểm mẫu ban đầu

Chương 2

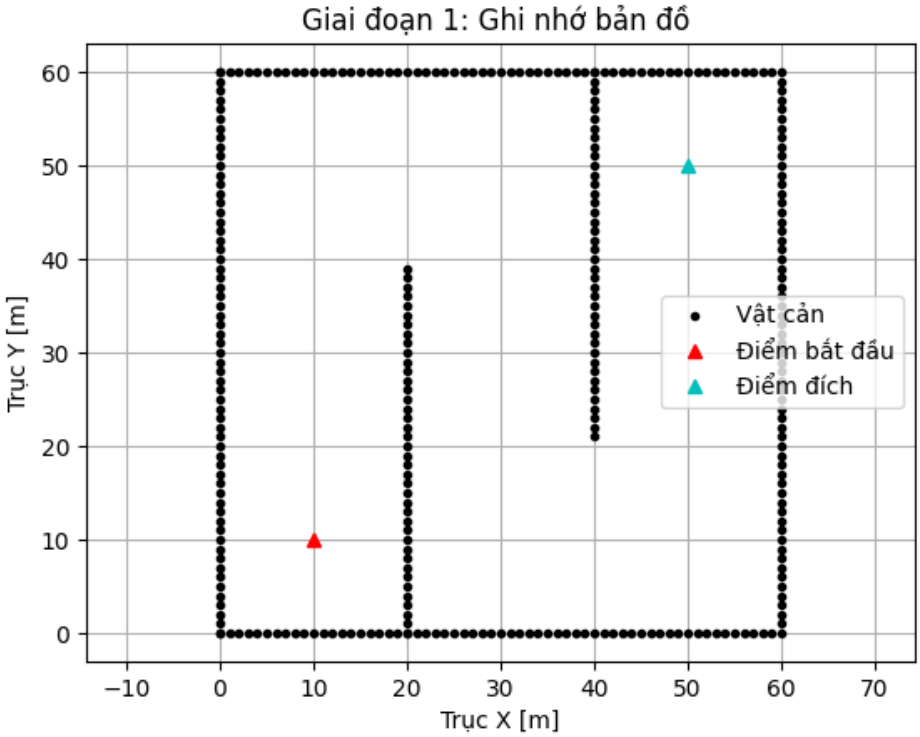
Triển khai thuật toán PRM

2.1 Các giai đoạn của thuật toán PRM

Thuật toán PRM được triển khai qua bốn giai đoạn chính, mỗi giai đoạn đóng vai trò quan trọng trong việc xây dựng bản đồ đường đi và tìm đường đi tối ưu.

2.1.1 Giai đoạn 1: Ghi nhớ bản đồ (vật cản)

Giai đoạn đầu tiên của thuật toán PRM là ghi nhớ bản đồ môi trường, bao gồm vị trí của các chướng ngại vật, điểm xuất phát và điểm đích. Đây là bước quan trọng để xác định không gian tự do mà robot có thể di chuyển.



Hình 2.1: Giai đoạn 1: Ghi nhớ bản đồ với vật cản, điểm bắt đầu và điểm đích

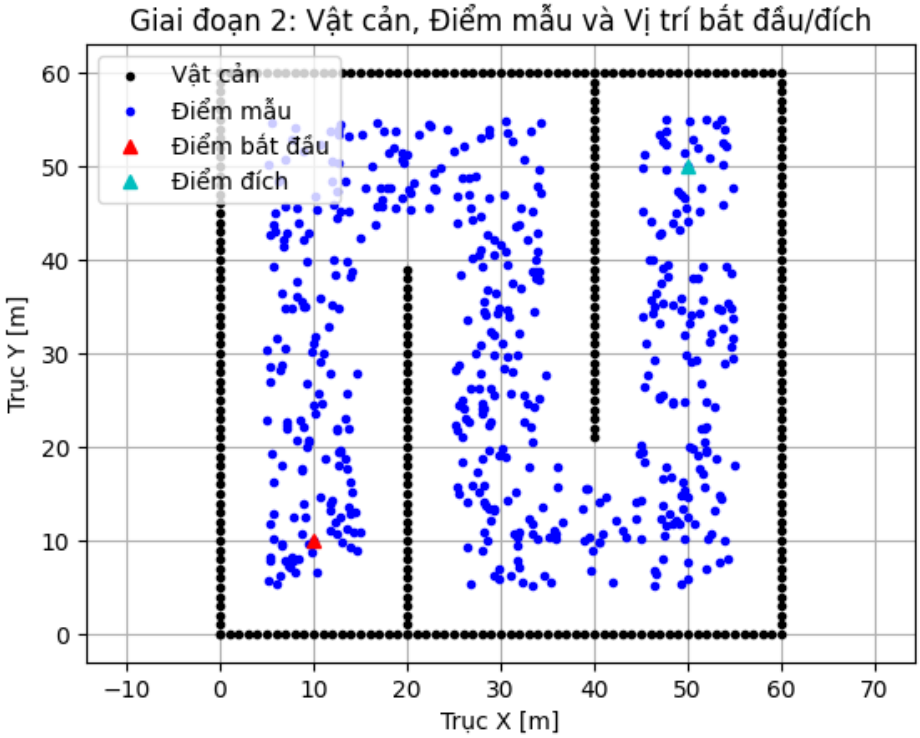
Trong hình 2.1, chúng ta có thể thấy:

- Các chấm đen biểu thị vị trí của các chướng ngại vật
- Tam giác đỏ biểu thị điểm xuất phát
- Tam giác xanh biểu thị điểm đích

Môi trường trong ví dụ này là một không gian 2D với các chướng ngại vật được bố trí theo hình dạng của một mê cung đơn giản. Robot cần tìm đường đi từ điểm xuất phát ở góc dưới bên trái đến điểm đích ở phía trên bên phải.

2.1.2 Giai đoạn 2: Tạo điểm mẫu

Sau khi đã ghi nhớ bản đồ môi trường, thuật toán PRM tiến hành sinh ngẫu nhiên các điểm mẫu trong không gian tự do. Các điểm này sẽ trở thành các đỉnh của đồ thị bản đồ đường đi.



Hình 2.2: Giai đoạn 2: Tạo điểm mẫu trong không gian tự do

Trong hình 2.2, chúng ta có thể thấy:

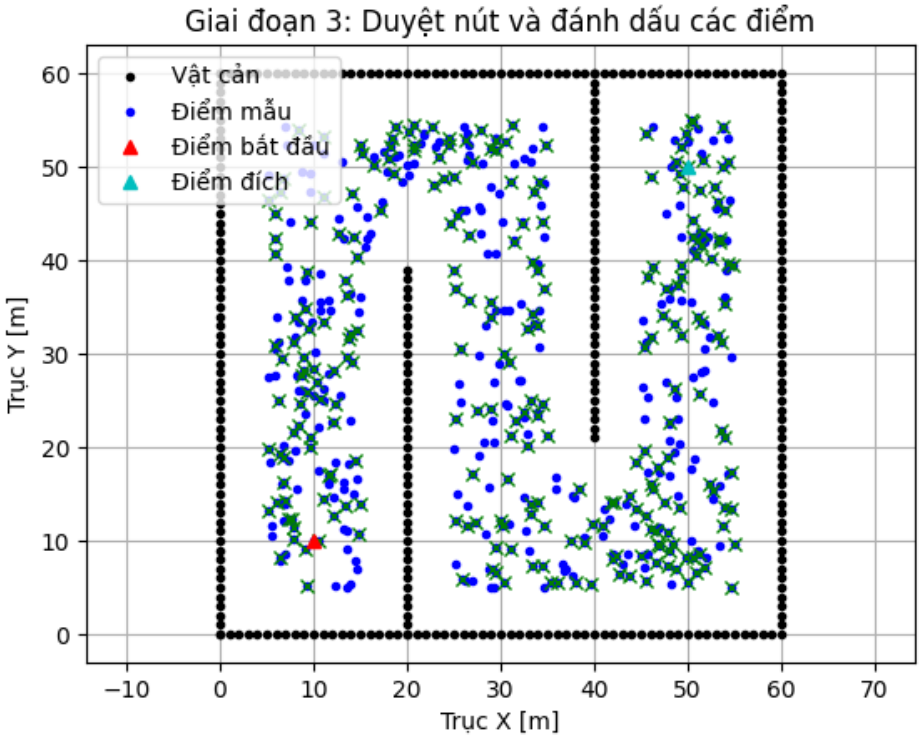
- Các chấm xanh biểu thị các điểm mẫu được sinh ngẫu nhiên trong không gian tự do
- Các điểm mẫu được phân bố đều trong toàn bộ không gian tự do, bao gồm cả các khu vực hẹp và rộng

Quá trình sinh điểm mẫu tuân theo một số nguyên tắc:

- Các điểm mẫu phải nằm trong không gian tự do, tức là không nằm trong hoặc quá gần các chướng ngại vật
- Số lượng điểm mẫu phải đủ lớn để đảm bảo khả năng tìm được đường đi, nhưng không quá lớn để tránh tốn kém về mặt tính toán
- Phân bố của các điểm mẫu nên đều trong không gian tự do để đảm bảo khả năng tìm được đường đi tối ưu

2.1.3 Giai đoạn 3: Duyệt nút và đánh dấu

Sau khi đã sinh các điểm mẫu, thuật toán PRM tiến hành kết nối các điểm này để tạo thành một đồ thị. Mỗi điểm sẽ được kết nối với một số điểm gần nhất nếu đường nối giữa chúng không đi qua chướng ngại vật.



Hình 2.3: Giai đoạn 3: Duyệt nút và đánh dấu các kết nối

Trong hình 2.3, chúng ta có thể thấy:

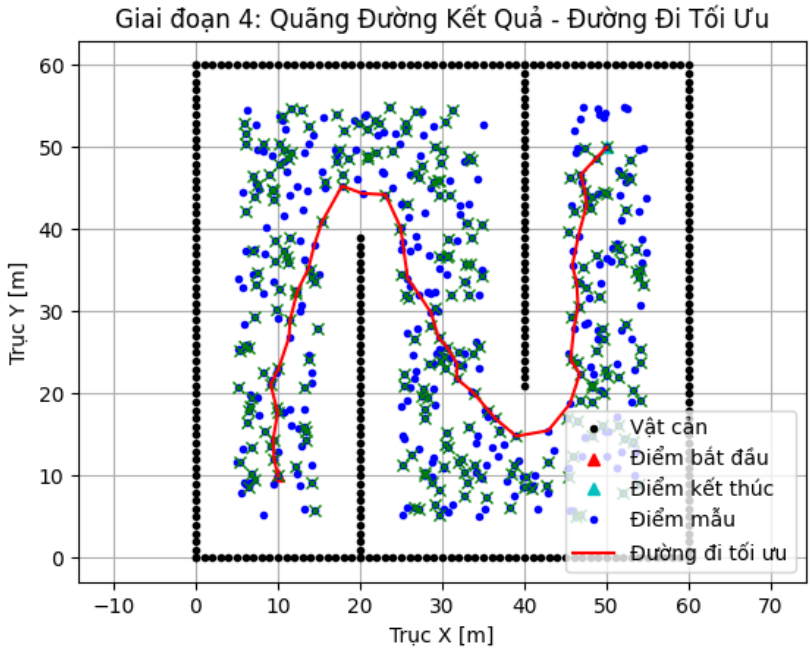
- Các chấm xanh vẫn biểu thị các điểm mẫu
- Các dấu "x" xanh lá biểu thị các kết nối giữa các điểm mẫu

Quá trình kết nối các điểm mẫu tuân theo một số nguyên tắc:

- Mỗi điểm sẽ được kết nối với K điểm gần nhất (K-Nearest Neighbors)
- Một kết nối chỉ được tạo ra nếu đường thẳng nối hai điểm không đi qua chướng ngại vật
- Độ dài của một kết nối không vượt quá một ngưỡng cho trước để tránh tạo ra các kết nối quá dài và không thực tế

2.1.4 Giai đoạn 4: Kết quả đường đi tối ưu

Giai đoạn cuối cùng của thuật toán PRM là tìm đường đi tối ưu từ điểm xuất phát đến điểm đích trên đồ thị đã xây dựng. Thuật toán sử dụng Dijkstra hoặc A* để tìm đường đi ngắn nhất.



Hình 2.4: Giai đoạn 4: Kết quả đường đi tối ưu

Trong hình 2.4, chúng ta có thể thấy:

- Đường màu đỏ biểu thị đường đi tối ưu từ điểm xuất phát đến điểm đích
- Đường đi này đi qua các điểm mẫu và tránh các chướng ngại vật

Đường đi tối ưu được tìm thấy bằng cách:

- Kết nối điểm xuất phát và điểm đích vào đồ thị bản đồ đường đi
- Sử dụng thuật toán Dijkstra để tìm đường đi ngắn nhất từ điểm xuất phát đến điểm đích
- Làm mịn đường đi nếu cần thiết để tạo ra một đường đi trơn tru và tự nhiên hơn

2.2 Cài đặt thuật toán PRM bằng Python

Dưới đây là cài đặt chi tiết của thuật toán PRM bằng ngôn ngữ Python, sử dụng các thư viện NumPy, Matplotlib và SciPy.

```python
import math
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial import KDTree

# Cau hinh co ban
N_SAMPLE = 500  # So diem lay mau
N_KNN = 10      # So hang xom gan nhat de ket noi
MAX_EDGE_LEN = 30.0  # Do dai cung toi da

show_animation = True  # Hien thi hoat hinh

# Lop Nut luu thong tin diem
class Node:
    def __init__(self, x, y, cost, parent_index):
        self.x = x  # Toa do x
        self.y = y  # Toa do y
        self.cost = cost  # Chi phi den diem nay
        self.parent_index = parent_index  # Chi so diem cha

    def __str__(self):
        return str(self.x) + "," + str(self.y) + "," + str(self.cost)
    + "," + str(self.parent_index)

# Ham chinh lap ke hoach duong di PRM
def prm_planning(start_x, start_y, goal_x, goal_y,
                 obstacle_x_list, obstacle_y_list, robot_radius):

    obstacle_kd_tree = KDTree(np.vstack((obstacle_x_list,
    obstacle_y_list)).T)  # Tao KDTree cho vat can

    sample_x, sample_y = sample_points(
        start_x, start_y, goal_x, goal_y,
        robot_radius,
        obstacle_x_list, obstacle_y_list,
        obstacle_kd_tree
    )  # Lay mau diem

    if show_animation:
        plt.plot(sample_x, sample_y, ".b")  # Ve diem lay mau

    road_map = generate_road_map(sample_x, sample_y, robot_radius,
    obstacle_kd_tree)  # Tao ban do duong di

    rx, ry = dijkstra_planning(start_x, start_y, goal_x, goal_y,
    road_map, sample_x, sample_y)  # Tim duong di ngan nhat

    return rx, ry
```

```
45
46  # Lay mau diem trong khong gian
47  def sample_points(sx, sy, gx, gy, rr, ox, oy, obstacle_kd_tree):
48      max_x = max(ox)  # Gioi han tren x
49      max_y = max(oy)  # Gioi han tren y
50      min_x = min(ox)  # Gioi han duoi x
51      min_y = min(oy)  # Gioi han duoi y
52
53      sample_x, sample_y = [], []
54      rng = np.random.default_rng()  # Tao so ngau nhien
55
56      while len(sample_x) <= N_SAMPLE:
57          tx = (rng.random() * (max_x - min_x)) + min_x  # Toa do x ngau
        nhien
58          ty = (rng.random() * (max_y - min_y)) + min_y  # Toa do y ngau
        nhien
59
60          dist, index = obstacle_kd_tree.query([tx, ty])  # Khoang cach
       den vat can
61
62          if dist >= rr:
63              sample_x.append(tx)
64              sample_y.append(ty)
65
66      sample_x.append(sx)  # Them diem bat dau x
67      sample_y.append(sy)  # Them diem bat dau y
68      sample_x.append(gx)  # Them diem ket thuc x
69      sample_y.append(gy)  # Them diem ket thuc y
70
71      return sample_x, sample_y
72
73  # Tao ban do duong di
74  def generate_road_map(sample_x, sample_y, rr, obstacle_kd_tree):
75      road_map = []
76      n_sample = len(sample_x)
77      sample_kd_tree = KDTree(np.vstack((sample_x, sample_y)).T)  # Tao
       KDTree cho diem mau
78
79      for (ix, iy) in zip(sample_x, sample_y):
80          dists, indexes = sample_kd_tree.query([ix, iy], k=n_sample)  #
        Tim hang xom gan nhat
81
82          edge_id = []
83
84          for ii in range(1, len(indexes)):
85              nx = sample_x[indexes[ii]]
86              ny = sample_y[indexes[ii]]
87
88              if not is_collision(ix, iy, nx, ny, rr, obstacle_kd_tree):
         # Kiem tra va cham
89                  edge_id.append(indexes[ii])
90
91              if len(edge_id) >= N_KNN:
92                  break
93
```

```
94          road_map.append(edge_id)
95
96      return road_map
97
98  # Kiem tra va cham giua hai diem
99  def is_collision(sx, sy, gx, gy, rr, obstacle_kd_tree):
100     x = sx
101     y = sy
102     dx = gx - sx
103     dy = gy - sy
104     yaw = math.atan2(gy - sy, gx - sx)  # Goc huong
105     d = math.hypot(dx, dy)  # Khoang cach
106
107     if d >= MAX_EDGE_LEN:
108         return True
109
110     D = rr
111     n_step = round(d / D)  # So buoc kiem tra
112
113     for i in range(n_step):
114         dist, index = obstacle_kd_tree.query([x, y])  # Khoang cach
    den vat can
115
116         if dist <= rr:
117             return True
118
119         x += D * math.cos(yaw)  # Di chuyen doc doan thang
120         y += D * math.sin(yaw)
121
122     dist, index = obstacle_kd_tree.query([gx, gy])  # Kiem tra diem
    cuoi
123
124     if dist <= rr:
125         return True
126
127     return False
128
129 # Tim duong di ngan nhat bang Dijkstra
130 def dijkstra_planning(sx, sy, gx, gy, road_map, sample_x, sample_y):
131     start_node = Node(sx, sy, 0.0, -1)  # Nut bat dau
132     goal_node = Node(gx, gy, 0.0, -1)  # Nut dich
133
134     open_set, closed_set = dict(), dict()
135     open_set[len(road_map) - 2] = start_node  # Them nut bat dau vao
    tap mo
136
137     path_found = True
138
139     while True:
140         if not open_set:
141             print("Cannot find path")  # Khong tim thay duong
142             path_found = False
143             break
144
```

```
145         c_id = min(open_set, key=lambda o: open_set[o].cost)  # Chon nut chi phi nho nhat
146         current = open_set[c_id]
147 
148         if show_animation and len(closed_set.keys()) % 2 == 0:
149             plt.gcf().canvas.mpl_connect(
150                 'key_release_event',
151                 lambda event: [exit(0) if event.key == 'escape' else None]
152             )  # Thoat khi nhan phim escape
153             plt.plot(current.x, current.y, "xg")  # Ve diem hien tai
154             plt.pause(0.001)
155 
156         if c_id == (len(road_map) - 1):
157             print("goal is found!")  # Tim thay dich
158             goal_node.parent_index = current.parent_index
159             goal_node.cost = current.cost
160             break
161 
162         del open_set[c_id]
163         closed_set[c_id] = current
164 
165         for i in range(len(road_map[c_id])):
166             n_id = road_map[c_id][i]
167             dx = sample_x[n_id] - current.x
168             dy = sample_y[n_id] - current.y
169             d = math.hypot(dx, dy)  # Khoang cach
170 
171             node = Node(sample_x[n_id], sample_y[n_id], current.cost + d, c_id)  # Tao nut moi
172             if n_id in closed_set:
173                 continue
174             if n_id in open_set:
175                 if open_set[n_id].cost > node.cost:
176                     open_set[n_id].cost = node.cost
177                     open_set[n_id].parent_index = c_id
178             else:
179                 open_set[n_id] = node
180 
181     if path_found is False:
182         return [], []
183 
184     rx, ry = [goal_node.x], [goal_node.y]  # Truy vet duong di
185     parent_index = goal_node.parent_index
186 
187     while parent_index != -1:
188         n = closed_set[parent_index]
189         rx.append(n.x)
190         ry.append(n.y)
191         parent_index = n.parent_index
192 
193     final_path_x = rx[::-1]  # Dao nguoc duong di x
194     final_path_y = ry[::-1]  # Dao nguoc duong di y
195 
```

```
196     path_str = " -> ".join(f"[{x:.2f}, {y:.2f}]" for x, y in zip(
    final_path_x, final_path_y))
197     print("The path found is: " + path_str)  # In duong di
198 
199     return rx, ry
200 
201 # Ham chinh chay chuong trinh
202 def main():
203     print("start!!")  # Bat dau
204 
205 # NOI NHAP TESTCASE
206     sx = 10.0  # Toa do x bat dau
207     sy = 10.0  # Toa do y bat dau
208     gx = 50.0  # Toa do x ket thuc
209     gy = 50.0  # Toa do y ket thuc
210     robot_size = 5.0  # Ban kinh robot
211 
212     ox = []  # Danh sach toa do x vat can
213     oy = []  # Danh sach toa do y vat can
214 
215     for i in range(60):
216         ox.append(float(i))
217         oy.append(0.0)  # Bien duoi
218 
219     for i in range(60):
220         ox.append(60.0)
221         oy.append(float(i))  # Bien phai
222 
223     for i in range(61):
224         ox.append(float(i))
225         oy.append(60.0)  # Bien tren
226 
227     for i in range(61):
228         ox.append(0.0)
229         oy.append(float(i))  # Bien trai
230 
231     for i in range(40):
232         ox.append(20.0)
233         oy.append(float(i))  # Tuong doc 1
234 
235     for i in range(40):
236         ox.append(40.0)
237         oy.append(60.0 - i)  # Tuong doc 2
238 # NOI KET THUC NHAP TESTCASE
239 
240     if show_animation:
241         plt.plot(ox, oy, ".k")  # Ve vat can
242         plt.plot(sx, sy, "^r")  # Ve diem bat dau
243         plt.plot(gx, gy, "^c")  # Ve diem ket thuc
244         plt.grid(True)
245         plt.axis("equal")
246 
247     rx, ry = prm_planning(sx, sy, gx, gy, ox, oy, robot_size)  # Lap
    ke hoach duong di
248 
```

```
249     assert rx, 'Cannot found path'  # Kiem tra duong di
250
251     if show_animation:
252         for i in range(len(rx) - 1, 0, -1):
253             plt.plot(rx[i - 1:i + 1], ry[i - 1:i + 1], "-r")  # Ve
    duong di
254             plt.pause(0.1)
255             plt.show()
256
257 if __name__ == '__main__':
258     main()
```

Listing 2.1: Cài đặt thuật toán PRM bằng Python

2.3 Phân tích chi tiết các hàm trong mã nguồn

Mã nguồn Python triển khai thuật toán PRM bao gồm nhiều hàm với các chức năng khác nhau. Dưới đây là phân tích chi tiết về các hàm chính trong mã nguồn.

2.3.1 Hàm prm_planning

Đây là hàm chính của thuật toán PRM, nhận vào các tham số như tọa độ điểm xuất phát, điểm đích, danh sách vật cản và bán kính robot. Hàm này điều phối toàn bộ quá trình lập kế hoạch đường đi, bao gồm:

- Tạo cây KDTree từ danh sách vật cản để tối ưu hóa việc tìm kiếm
- Gọi hàm sample_points để sinh các điểm mẫu trong không gian tự do
- Gọi hàm generate_road_map để tạo bản đồ đường đi từ các điểm mẫu
- Gọi hàm dijkstra_planning để tìm đường đi tối ưu từ điểm xuất phát đến điểm đích
- Trả về danh sách các điểm trên đường đi tối ưu

2.3.2 Hàm sample_points

Hàm này sinh ngẫu nhiên các điểm mẫu trong không gian tự do. Các điểm này sẽ trở thành các đỉnh của đồ thị bản đồ đường đi. Hàm này:

- Xác định giới hạn của không gian (min_x, max_x, min_y, max_y)
- Sinh ngẫu nhiên các điểm trong không gian
- Kiểm tra xem điểm có nằm trong không gian tự do hay không bằng cách tính khoảng cách đến vật cản gần nhất
- Thêm điểm xuất phát và điểm đích vào danh sách điểm mẫu
- Trả về danh sách các điểm mẫu

2.3.3 Hàm generate_road_map

Hàm này tạo bản đồ đường đi từ các điểm mẫu bằng cách kết nối các điểm gần nhau nếu đường nối giữa chúng không đi qua vật cản. Hàm này:

- Tạo cây KDTree từ các điểm mẫu để tối ưu hóa việc tìm kiếm điểm gần nhất
- Với mỗi điểm mẫu, tìm K điểm gần nhất
- Kiểm tra xem đường nối giữa điểm hiện tại và điểm gần nhất có đi qua vật cản hay không bằng hàm is_collision
- Nếu không đi qua vật cản, thêm kết nối vào bản đồ đường đi
- Trả về bản đồ đường đi dưới dạng danh sách các kết nối

2.3.4 Hàm is_collision

Hàm này kiểm tra xem đoạn đường từ điểm bắt đầu đến điểm kết thúc có va chạm với vật cản hay không. Hàm này:

- Tính toán hướng và khoảng cách giữa hai điểm
- Kiểm tra xem khoảng cách có vượt quá ngưỡng cho phép hay không
- Chia đoạn đường thành nhiều đoạn nhỏ và kiểm tra từng đoạn

- Với mỗi đoạn, tính khoảng cách đến vật cản gần nhất và so sánh với bán kính robot
- Trả về True nếu có va chạm, False nếu không có va chạm

2.3.5 Hàm dijkstra_planning

Hàm này sử dụng thuật toán Dijkstra để tìm đường đi tối ưu từ điểm xuất phát đến điểm đích trên đồ thị bản đồ đường đi. Hàm này:

- Khởi tạo các tập hợp open_set và closed_set
- Thêm điểm xuất phát vào open_set
- Lặp cho đến khi tìm thấy đường đi hoặc không còn điểm nào trong open_set
- Trong mỗi vòng lặp, chọn điểm có chi phí thấp nhất từ open_set
- Nếu điểm hiện tại là điểm đích, kết thúc thuật toán
- Nếu không, thêm điểm hiện tại vào closed_set và xét các điểm kề
- Với mỗi điểm kề, tính chi phí mới và cập nhật nếu chi phí mới thấp hơn chi phí hiện tại
- Sau khi tìm thấy đường đi, truy vết từ điểm đích về điểm xuất phát để xây dựng đường đi
- Trả về danh sách các điểm trên đường đi

Chương 3

Ứng dụng thực tế của thuật toán PRM

3.1 Robot di chuyển trong nhà kho

Một trong những ứng dụng phổ biến nhất của thuật toán PRM là trong lĩnh vực robot di chuyển trong nhà kho. Các robot tự động được sử dụng để vận chuyển hàng hóa, sắp xếp kho bãi, và thực hiện các nhiệm vụ logistics khác.

Thuật toán PRM giúp robot tìm đường đi tối ưu trong môi trường nhà kho phức tạp với nhiều kệ hàng, hàng hóa, và các chướng ngại vật khác. Bằng cách xây dựng một bản đồ đường đi trước, robot có thể di chuyển nhanh chóng và an toàn, tránh va chạm với các chướng ngại vật và tối ưu hóa thời gian di chuyển.

Một số ưu điểm của việc sử dụng thuật toán PRM trong robot nhà kho:

- Khả năng xử lý môi trường phức tạp với nhiều chướng ngại vật
- Tính toán đường đi nhanh chóng, phù hợp với yêu cầu thời gian thực
- Khả năng tái sử dụng bản đồ đường đi cho nhiều truy vấn khác nhau
- Dễ dàng mở rộng và tùy chỉnh cho các yêu cầu cụ thể của nhà kho

3.2 Lập kế hoạch đường bay cho UAV

Thuật toán PRM cũng được ứng dụng rộng rãi trong lĩnh vực lập kế hoạch đường bay cho UAV (Unmanned Aerial Vehicle) hay còn gọi là drone. Trong môi

trường đô thị với nhiều tòa nhà cao tầng, cây cối, và các chướng ngại vật khác, việc tìm đường bay an toàn và hiệu quả là một thách thức lớn.

Thuật toán PRM giúp UAV tìm đường bay tối ưu, tránh va chạm với các chướng ngại vật, và tối ưu hóa tiêu thụ năng lượng. Bằng cách xây dựng một bản đồ đường đi trong không gian 3D, UAV có thể di chuyển an toàn và hiệu quả trong môi trường phức tạp.

Một số ưu điểm của việc sử dụng thuật toán PRM trong lập kế hoạch đường bay cho UAV:

- Khả năng xử lý không gian 3D phức tạp
- Tính toán đường bay tối ưu về mặt khoảng cách và tiêu thụ năng lượng
- Khả năng tránh các khu vực cấm bay hoặc nguy hiểm
- Dễ dàng tích hợp với các hệ thống điều khiển bay tự động

3.3 Hỗ trợ phẫu thuật robot

Một ứng dụng đặc biệt quan trọng của thuật toán PRM là trong lĩnh vực phẫu thuật robot. Trong phẫu thuật, robot cần di chuyển các dụng cụ phẫu thuật một cách chính xác và an toàn, tránh va chạm với các cơ quan nội tạng và mô.

Thuật toán PRM giúp robot phẫu thuật tìm đường đi tối ưu cho các dụng cụ phẫu thuật, đảm bảo an toàn và hiệu quả. Bằng cách xây dựng một bản đồ đường đi trong không gian phẫu thuật, robot có thể thực hiện các thao tác phẫu thuật một cách chính xác và an toàn.

Một số ưu điểm của việc sử dụng thuật toán PRM trong phẫu thuật robot:

- Độ chính xác cao, đảm bảo an toàn cho bệnh nhân
- Khả năng xử lý không gian phẫu thuật phức tạp
- Tính toán đường đi nhanh chóng, phù hợp với yêu cầu thời gian thực
- Dễ dàng tích hợp với các hệ thống phẫu thuật robot hiện đại

Chương 4

Kết luận và hướng phát triển

4.1 Tóm tắt về thuật toán PRM

Thuật toán PRM (Probabilistic Roadmap) là một phương pháp lập kế hoạch đường đi dựa trên xác suất, được sử dụng rộng rãi trong lĩnh vực robotics. Thuật toán này hoạt động bằng cách sinh ngẫu nhiên các điểm trong không gian tự do, kết nối chúng lại với nhau để tạo thành một bản đồ đường đi, và sử dụng các thuật toán tìm đường như Dijkstra hoặc A* để tìm đường đi tối ưu từ điểm xuất phát đến điểm đích.

Thuật toán PRM có nhiều ưu điểm nổi bật:

- Hiệu quả trong không gian có nhiều chiều
- Khả năng xử lý môi trường phức tạp với nhiều chướng ngại vật
- Giai đoạn xây dựng bản đồ chỉ cần thực hiện một lần, sau đó có thể sử dụng lại cho nhiều truy vấn khác nhau
- Dễ dàng mở rộng và tùy chỉnh cho các ứng dụng cụ thể

Tuy nhiên, thuật toán PRM cũng có một số hạn chế:

- Hiệu suất giảm mạnh trong không gian nhiều chiều
- Khó xử lý các chướng ngại vật di động hoặc thay đổi hình dạng
- Phụ thuộc nhiều vào chất lượng phân bố điểm mẫu ban đầu

4.2 Hướng phát triển và cải tiến

Mặc dù thuật toán PRM đã chứng minh hiệu quả trong nhiều ứng dụng thực tế, vẫn còn nhiều hướng phát triển và cải tiến để nâng cao hiệu suất và khả năng ứng dụng của thuật toán này.

4.2.1 Cải tiến phương pháp sinh điểm mẫu

Một trong những yếu tố quan trọng ảnh hưởng đến hiệu suất của thuật toán PRM là chất lượng của các điểm mẫu. Các phương pháp sinh điểm mẫu thông minh hơn có thể giúp cải thiện hiệu suất của thuật toán, đặc biệt trong không gian nhiều chiều. Một số hướng cải tiến bao gồm:

- Sử dụng các phương pháp sinh điểm mẫu dựa trên đặc điểm của không gian, ví dụ như tập trung nhiều điểm mẫu hơn ở các khu vực hẹp hoặc phức tạp
- Áp dụng các phương pháp học máy để dự đoán vị trí tốt cho các điểm mẫu
- Sử dụng các phương pháp sinh điểm mẫu thích ứng, điều chỉnh dựa trên kết quả của các lần sinh trước đó

4.2.2 Xử lý môi trường động

Một hạn chế lớn của thuật toán PRM là khó xử lý các môi trường động với các chướng ngại vật di động hoặc thay đổi hình dạng. Một số hướng cải tiến để xử lý môi trường động bao gồm:

- Phát triển các phiên bản động của thuật toán PRM, có khả năng cập nhật bản đồ đường đi khi môi trường thay đổi
- Kết hợp với các phương pháp dự đoán chuyển động để dự đoán vị trí của các chướng ngại vật di động
- Sử dụng các phương pháp lập kế hoạch đường đi thời gian thực để điều chỉnh đường đi khi phát hiện các thay đổi trong môi trường

4.2.3 Tối ưu hóa hiệu suất tính toán

Hiệu suất tính toán là một yếu tố quan trọng trong các ứng dụng thời gian thực. Một số hướng cải tiến để tối ưu hóa hiệu suất tính toán của thuật toán PRM bao gồm:

- Sử dụng các cấu trúc dữ liệu hiệu quả hơn để lưu trữ và truy vấn bản đồ đường đi
- Áp dụng các phương pháp song song hóa để tận dụng sức mạnh của các hệ thống đa nhân
- Phát triển các phiên bản xấp xỉ của thuật toán PRM, đánh đổi một phần độ chính xác để đạt được hiệu suất tính toán cao hơn

4.2.4 Kết hợp với các phương pháp học máy

Một hướng phát triển đầy hứa hẹn là kết hợp thuật toán PRM với các phương pháp học máy để nâng cao hiệu suất và khả năng thích ứng. Một số hướng kết hợp bao gồm:

- Sử dụng học tăng cường để tối ưu hóa các tham số của thuật toán PRM
- Áp dụng học sâu để dự đoán các khu vực có khả năng cao chứa đường đi tối ưu
- Kết hợp với các phương pháp học không giám sát để phát hiện các mẫu và cấu trúc trong không gian

4.3 Kết luận

Thuật toán PRM là một công cụ mạnh mẽ trong lĩnh vực lập kế hoạch đường đi cho robot, với nhiều ứng dụng thực tế trong các lĩnh vực như robot di chuyển trong nhà kho, lập kế hoạch đường bay cho UAV, và hỗ trợ phẫu thuật robot. Mặc dù còn một số hạn chế, thuật toán PRM vẫn là một trong những phương

pháp hiệu quả nhất để giải quyết bài toán lập kế hoạch đường đi trong môi trường phức tạp.

Với sự phát triển không ngừng của công nghệ và các phương pháp tính toán, chúng ta có thể kỳ vọng vào những cải tiến đáng kể trong tương lai, giúp thuật toán PRM trở nên hiệu quả hơn, linh hoạt hơn, và có khả năng ứng dụng rộng rãi hơn trong nhiều lĩnh vực khác nhau.

Tài liệu tham khảo

[1] Kavraki, L. E., Svestka, P., Latombe, J. C., & Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation, 12(4), 566-580.

[2] LaValle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning. Technical Report, Computer Science Department, Iowa State University.

[3] Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. The International Journal of Robotics Research, 30(7), 846-894.

[4] Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L. E., & Thrun, S. (2005). Principles of robot motion: theory, algorithms, and implementations. MIT press.

[5] Amato, N. M., & Wu, Y. (1996). A randomized roadmap method for path and manipulation planning. In Proceedings of IEEE International Conference on Robotics and Automation (Vol. 1, pp. 113-120).

[6] Atsushi Sakai. *PythonRobotics: ProbabilisticRoadMap*. GitHub repository. `https://github.com/AtsushiSakai/PythonRobotics/tree/master/PathPlanning/ProbabilisticRoadMap`