BẢN ĐỒ HÀNH LANG TƯỜNG MINH: BẢN ĐỒ CHO BÀI TOÁN TÌM ĐƯỜNG ĐA ĐỐI TƯỢNG

EXPLICIT CORRIDOR MAP: ROADMAP FOR MULTIPLE MOVING OBJECTS PATH FINDING PROBLEM

[1]Trần Nhật Hoàng Anh, [2]Vương Bá Thịnh
[1]Đại học Giao thông vận tải Thành phố Hồ Chí Minh
[2]Đại học Bách Khoa – ĐHQG Thành phố Hồ Chí Minh

Tóm tắt: *Tìm đường đi từ vị trí A đến vị trí B, đồng thời tránh vật cản và phản ứng với sự thay đổi của môi trường xung quanh là một bài toán lớn và cổ điển, đặc biệt là không dễ đối với rô bốt. Tìm đường đối với rô bốt chỉ có thể tính toán một khi bản đồ cho môi trường đã được xác định. Trong bài báo này, nhóm tác giả trình bày cách xây dựng bản đồ hành lang tường minh (Explicit Corridor Map (ECM)) cho môi trường có chứa các vật cản và bản đồ có thể dễ dàng được cập nhật khi môi trường có sự thay đổi. Với một môi trường phẳng chứa n vật cản, độ phức tạp về mặt lưu trữ là $O(n)$ và tiêu tốn thời gian tính toán là $O(n \log n)$. Kết quả thực nghiệm cho thấy ECM tính toán rất nhanh trên môi trường 2D, đồng thời việc tìm đường trên bản đồ này chỉ trong khoảng vài ms. Bản đồ hành lang tường minh cũng phù hợp với việc mô phỏng đám đông di chuyển.*

Từ khóa: *Lưới điều hướng, tìm đường.*

Chỉ số phân loại: *2.2*

Abstract: *Path planning from position A to position B, while avoiding obstacles and responding to changes in the surrounding environment is a big and fundamental task, especially uneasy for rô bốts. Path finding for rô bốts can only be calculated once the roadmap for the environment has been identified. In this paper, we present a way to build for the environment that contains obstructions an Explicit Corridor Map (ECM), which can be simply updated as the environment changes. For a planar environment with n obstacle vertices, storage complexity is $O(n)$ and computational time consumption is $O(n \log n)$. Experimental results showed that ECM is excuted very fast in large 2D environments; simultaneously, it can be used to compute paths within milliseconds. This enables simulations for moving crowds.*

Keywords: *Navigation mesh, path finding.*

Classification number: *2.2*

1. Giới thiệu

Thiết bị tự hành hay người máy là một thiết bị nhân tạo trông giống như con người, hoạt động tự động hoặc bán tự động bằng sự điều khiển của máy tính hay các vi mạch điện tử được lập trình. Chúng có nhiều ứng dụng hữu ích trong các lĩnh vực sản xuất, thám hiểm vụ trụ… hoặc để phục vụ cho mục đích trinh thám quân sự, dân sự. Các ứng dụng này làm phát sinh hai vấn đề chính trong lĩnh vực Robotics: Lập kế hoạch chuyển động và truy vết hành trình. Trong bài toán chính thứ nhất, các thiết bị được đưa về là một rô bốt điểm, quy về bài toán chung tìm đường cho đối tượng. Đối tượng cần di chuyển mượt mà và tránh va chạm với các vật cản cũng như các đối tượng khác. Đối tượng phải di chuyển trong một không gian gọi là không gian di chuyển được.

Lưới điều hướng (Navigation Mesh) là một dạng không gian di chuyển được. Nó có thể là một lưới các ô vuông liên kết với nhau, hoặc các đa giác lồi (thường là tam giác).

Trong bài báo, nhóm tác giả trình bày bản đồ hành lang tường minh là một dạng lưới điều hướng. Với độ phức tạp tính toán là là $O(n \log n)$ và chỉ phụ thuộc vào độ phức tạp của môi trường, trong khi đó phương pháp lưới ô phụ thuộc vào kích thước môi trường (không hiệu quả khi môi trường lớn). Đồng thời phương pháp ECM tạo ra đồ thị thưa

($O(n)$ space) cho nên việc tìm đường sẽ dễ dàng và nhanh chóng. Nó cũng hỗ trợ tác vụ như tìm kiếm xem có bao nhiêu vật cản gần nhất. Và có thể cập nhật bản đồ trong thời gian thực khi thêm hoặc xóa vật cản.

2. Công trình liên quan

2.1. Không gian tìm đường

Vấn đề tìm đường có thể nói là được bắt nguồn từ các nghiên cứu về rô bốt. Nơi mà, các nghiên cứu viên phải tính toán một một quỹ đạo không có va chạm (Collision - free) từ một cấu hình ban đầu đến một cấu hình khác. Ví dụ, như một cánh tay rô bốt với các khớp quay, một cấu hình có thể là tập vị trí của các khớp quay.

Một tập hợp tất cả các cấu hình – $\mathcal{C}$ sẽ được chia thành hai phần, không có va chạm ($\mathcal{C}_{free}$) và phần còn lại ($\mathcal{C}_{obs}$). Trong $\mathcal{C}_{free}$, rô bốt sẽ không chạm bất kì vật cản nào. Tại các cấu hình 3D, ví dụ như các rô bốt di chuyển trong không gian 3D, những cấu hình này quá phức tạp để miêu tả. Vì vậy, một cách giải quyết thông thường là miêu tả không gian dưới dạng lấy mẫu. Các giải pháp nổi tiếng của kỹ thuật này là Probabilistic Roadmaps (PRMs) [1] và Rapidly - Exploring Random Trees (RRTs) [2].

Trong mô phỏng đám đông, không gian của các đối tượng thường là không gian ba chiều. Tuy nhiên, các đối tượng thường được giới hạn trong các bề mặt di chuyển được. (bề mặt di chuyển được là $\mathcal{C}_{free}$). Đối với mô phỏng đám đông, các bề mặt di chuyển được thường được chiếu xuống một mặt phẳng *P* mà không gây ra bất kỳ sự chồng chéo nào. Toll và các cộng sự [3] mặc định các bề mặt di chuyển được được chiếu xuống mặt phẳng P – là không gian hai chiều.

2.2. Lưới điều hướng

Có rất nhiều cách để tự động chia $\mathcal{C}_{free}$ thành các thành phần liên kết với nhau. Snook [4] và Tozour [5] là những người đầu tiên sử dụng định nghĩa lưới điều hướng mà hiện tại trở nên rất thông dụng trong lĩnh vực tìm đường và mô phỏng đám đông.

Thuật ngữ "lưới điều hướng" (Navigation Mesh) về cơ bản biểu diễn một cách phân chia không gian để sử dùng cho mục đích điều hướng. Có rất nhiều loại lưới điều hướng, như là *Trapezoidal Map* (chia $\mathcal{C}_{free}$ thành các tấm theo chiều dọc tại mỗi đỉnh của các vật cản), hay là *Triangulation* (chia $\mathcal{C}_{free}$ thành các hình tam giác trong đó các đỉnh của tam giác là đỉnh của vật cản) và *Grid* (chia $\mathcal{C}_{free}$ thành các hình vuông),... Hình 1 mô tả ba loại lưới điều hướng.

(a) Trapezoidal Map *(b) Triangulation* *(c) Grid*

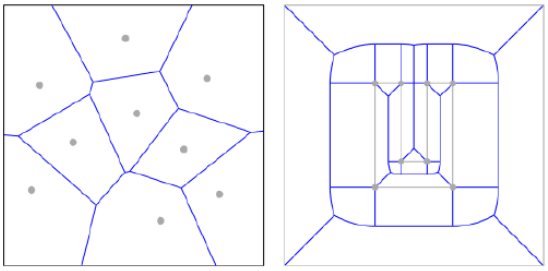
Hình 1. *Các phương pháp phân chia không gian.*

2.3. Sơ đồ Voronoi

Sơ đồ Voronoi (Voronoi Diagram - VD) là một khái niệm căn bản về cấu trúc dữ liệu hình học với nhiều ứng dụng liên quan.

Trong trường hợp đơn giản nhất, khi các vật thể là các điểm, ta có một tập hợp *s* gồm nhiều điểm trên mặt phẳng, được gọi là điểm Voronoi. Mỗi điểm *s* ứng với một ô Voronoi, hay còn gọi là ô Dirichlet, ký hiệu là V(*s*), bao gồm tất cả các điểm gần *s* hơn tất cả các điểm Voronoi khác. Các cạnh của sơ đồ Voronoi là tập các điểm có khoảng cách tới hai điểm Voronoi gần nhất là như nhau. Các đỉnh của sơ đồ Voronoi là các điểm có khoảng cách tới ít nhất ba điểm Voronoi gần nhất là như nhau. Hình 2a là một ví dụ của sơ đồ Voronoi. Đồ thị đối ngẫu của sơ đồ Voronoi là đồ thị có mỗi điểm cho mỗi ô của VD và các cạnh cho mỗi cặp ô mà có chung cạnh trong VD. Đồ thị này còn có tên là Delaunay Triangulation (DT). Sơ đồ Voronoi có thể mở rộng cho các đoạn thẳng và đa giác. Phiên bản này thường được gọi là sơ đồ Voronoi tổng quát (Generalized Voronoi Diagram hoặc GVD). Hình 2b là một ví dụ của sơ đồ Voronoi tổng quát.

Có rất nhiều cách dùng để tính toán sơ đồ Voronoi trong thời gian $(n \log n)$, bao gồm thuật toán mặt phẳng quét của Fortune [6], giải thuật theo hướng chia để trị của Shamos và Hoey [7], xây dựng gia tăng của Green và Sibson [8]. Ngoài ra, GVD có thể được tính toán gần đúng bằng giải thuật sử dụng đồ họa của Hoff et al [9].



(a) Sơ đồ Voronoi *(b) GVD*

Hình 2. *Sơ đồ Voronoi.*

3. Bản đồ hành lang tường minh

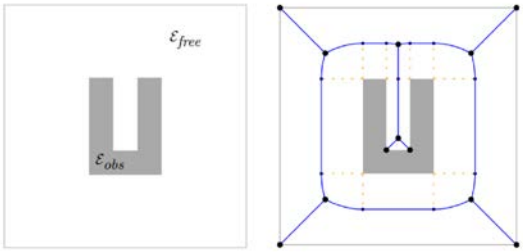
➢ Môi trường 2D

Một môi trường 2D $\mathscr{E}$ là tập con giới hạn của mặt phẳng 2D với các vật cản hình đa giác khép kín. Không gian vật cản $\mathscr{E}_{\text{obs}}$ là tập hợp của tất cả vật cản bao gồm cả đường biên của môi trường. Phần còn lại trong môi trường là không gian trống $\mathscr{E}_{free}$. Một minh họa của môi trường 2D được hiển thị trong hình 3a. Cho n là số đỉnh để định nghĩa $\mathscr{E}_{\text{obs}}$ bằng cách dùng các hình đa giác đơn giản, đường thẳng, điểm. Chúng ta gọi n là độ phức tạp của môi trường.

➢ Trục trung gian

Trục trung gian (Medial Axis) làm một tập con của sơ đồ Voronoi tổng quát (GVD), nó không có các cạnh và các đỉnh nằm trong vật cản. Mỗi cung của trục trung gian A là đường phân chia được tạo ra bởi hai nguồn: điểm hoặc đoạn thẳng của $\mathscr{E}_{free}$. Nếu một nguồn là một đoạn thẳng và nguồn còn lại là một điểm thì A sẽ là đường cong; ngược lại A là đường thẳng.

Các đỉnh có bậc bằng 1, 3, hoặc cao hơn là đỉnh chính. Các đỉnh có bậc bằng 2 sẽ là đỉnh phụ, bởi vì trục trung gian chỉ thay đổi hình dạng tại các đỉnh này. Một cạnh của trục trung gian là một chuỗi các cung trục trung gian giữa hai đỉnh chính. Trong hình 3b cho thấy trục trung gian của một môi trường 2D đơn giản chỉ gồm một vật cản hình chữ U và biên của môi trường vuông.



(a) Môi trường *(b) Trục trung gian*

Hình 3. *Môi trường 2D đơn giản.*

➢ Bản đồ hành lang tường minh

Bản đồ hành lang tường minh (Explitcit Corridor Map) là một biểu diễn đồ thị của trục trung gian với thông tin về vật cản gần nhất. Nó mô tả mỗi cung của trục trung gian và không gian trống xung quanh một cách hiệu quả. Như vậy, nó là một lưới điều hướng nhỏ gọn để tìm đường đi cho bất cứ nhân vật có bán kính bất kì.

Cho một môi trường 2D $\mathscr{E}$ với các vật cản, bản đồ hành lang tường minh ($\mathscr{E}$) là một biểu diễn mở rộng của trục trung gian ($\mathscr{E}$) bằng một đồ thị vô hướng $G = (V, E)$.

Trong đó:

- V là tập các đỉnh chính của trục trung gian;
- E là tập các cạnh của trục trung gian;
- Mỗi cạnh $e_{ij} \in E$ biểu diễn cung trục trung gian giữa hai đỉnh chính $v_{i,j} \in V$. Nó được biểu diễn bằng một chuỗi $n' \geq 2$ điểm uốn (bending point) $bp_0,\ldots,bp_{n'-1}$ trong đó $bp_0 = v_i$, $bp_{n'-1} = v_j$ và $bp_1,\ldots,bp_{n'-2}$ là các đỉnh phụ còn lại trên cạnh;
- Mỗi điểm uốn là một đỉnh của trục trung gian kết hợp với thông tin về vật cản gần nhất. Một điểm uốn bp_k trên một cạnh lưu trữ hai điểm vật cản gần nhất l_k và r_k ở bên trái và bên phải cạnh.

➢ Bản đồ hành lang tường minh trong môi trường động

Môi trường động là môi trường trong đó vật cản có thể xuất hiện, biến mất, hoặc di chuyển. Những vật cản động có ảnh hưởng nhiều đến môi trường. Trong nhiều trường hợp, phương pháp tránh va chạm cục bộ không thể dẫn đường cho đối tượng tới đích, đối tượng có thể bị kẹt ở đường đi cũ, phải tìm đường đi mới.

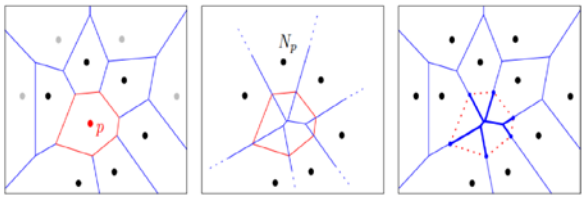
Khi cho một điểm, chúng ta có thể thực hiện một truy vấn **Point - Location** để tìm ra ô ECM nào chứa điểm này. Chúng ta có thể sử dụng giải thuật Point - Location của Kirkpatrick [10] có thể trả lời được câu truy vấn này trong thời gian $O(n \log n)$ và cần $O(n)$ không gian bộ nhớ.

Với tìm tập vật cản Np mà tạo ra cạnh Voronoi với p, tính sơ đồ sơ đồ Voronoi của các điểm vật cản, một điểm vật cản p có thể được xóa như sau: Voronoi (VD) của Np từ đầu, và thay những cạnh Voronoi cũ của p bằng những cạnh của VD mới này. Điều này được minh họa ở hình 4.

Ý tưởng tương tự cho xóa một điểm, đoạn thẳng, đa giác lồi trong ECM. Đặt O là vật cản cần xóa và giả sử nó không giao với các vật cản khác, như vậy sẽ có một vòng khép kín các cạnh ECM bao quanh O. Đặt Z_O là khu vực bao bởi vòng cạnh này. Chúng ta chỉ cần cập nhập ECM trong (và trên biên của) Z_O. Hơn nữa, để tính ECM mới trong Z_O, chúng ta chỉ cần phần các vật cản mà sinh ra cạnh ECM với O.

Đặt m là số ô ECM trên cạnh xung quanh O. Tập các vật cản lân cận N_O của O có kích thước (m), nó có thể tìm thấy trong thời gian (m) khi duyệt cạnh ECM xung quanh O. Tính ECM cho N_O tốn thời gian $(m \log m)$. Kết hợp với truy vấn Point - Location lúc bắt đầu, giải thuật hoàn chỉnh tốn thời gian $(log\ n + m\ log\ m)$. Nếu O có con trỏ chỉ đến các cạnh xung quanh, thì không cần truy vấn point-location, do đó phần $log\ n$ có thể loại bỏ.

Ý tưởng tương tự khi thêm một vật cản O. Ngoại trừ việc khi cập nhật ECM, chúng ta chỉ phải thêm chính vật cản đó vào N_O, so với việc xóa vật cản phải xây dựng trục trung gian cho tất cả các vật cản lân cận của vật cản đang bị xóa. Do đó thời gian thêm vật cản sẽ nhanh hơn một chút so với khi xóa.



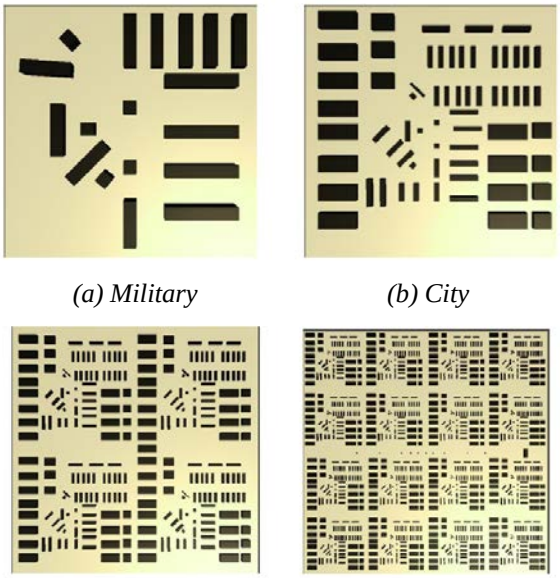
(a) Trước khi xóa *(b) VD của Np* *(c) Sau khi xóa*

Hình 4. *Xóa một điểm vật cản p từ sơ đồ Voronoi.*

4. Kết quả thực nghiệm

Trong phần này, nhóm nghiên cứu tiến hành đo hiệu năng cũng như tính toán thời gian một số tính năng như xây dựng EMC, thêm hoặc xóa vật cản.

Việc tính toán được đo đạc trên máy có thông số cấu hình như sau: Win 10, CPU i7 7500U, ram 4GB. Ngoài ra, tất cả những số liệu về thời gian mà nhóm nghiên cứu có đều được tính trung bình thông qua 10 lần đo.



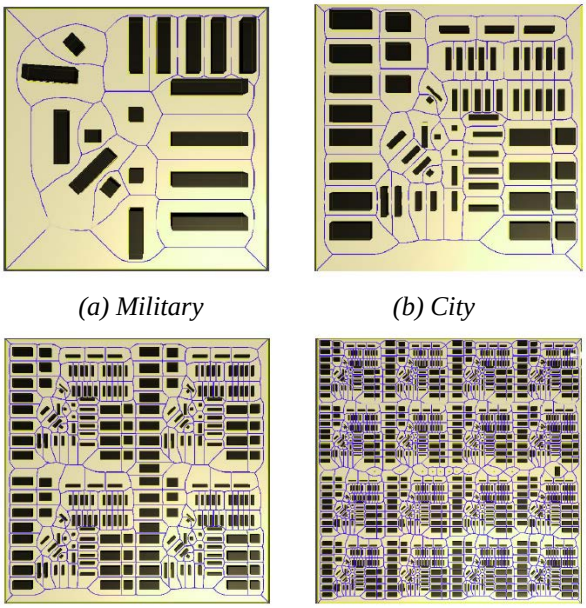
(a) Military *(b) City*

(c) City 2x2 *(d) City 4x4*

Hình 5. *Các môi trường thực nghiệm.*

Nhóm nghiên cứu đã thử nghiệm trên 4 môi trường ảo có kích thước khác nhau: Military, City, City 2x2, City 4x4 (xem hình 5). Military là một môi trường đơn giản với số lượng nhỏ vật cản và có kích thước (size) trên Unity là 200 x 200. City là một môi trường

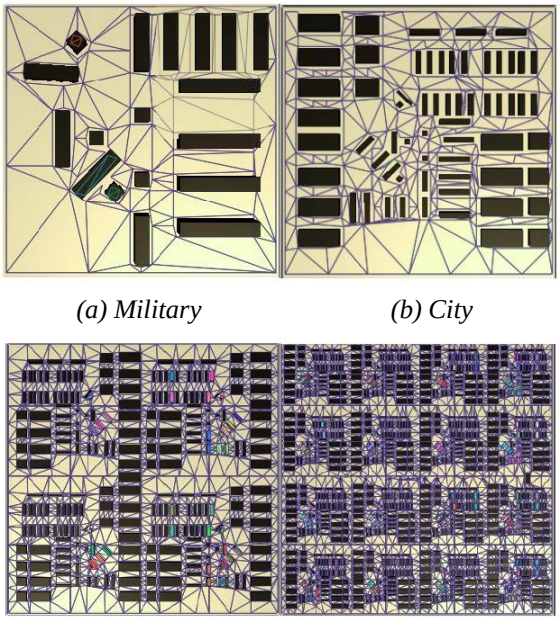
thành phố ảo phức tạp hơn, số lượng vật cản nhiều hơn và có kích thước 500 x 500. City 2x2 và City 4x4 tương tự với như City với kích thước lần lượt là 1000 x 1000 và 2000 x 2000.



(a) Military *(b) City*

(c) City 2x2 *(d) City 4x4*

Hình 6. *ECM sau khi xây dựng xong.*



(a) Military *(b) City*

(c) City 2x2 *(d) City 4x4*

Hình 7. *Recast sau khi xây dựng xong.*

Trong phần này nhóm nghiên cứu sẽ so sánh thời gian xây dựng ECM với Recast (hiện thực bởi Aron Granberg [11]). Recast là một lưới điều hướng rất phổ biến trong việc phát triển trò chơi, một phiên bản tùy chỉnh lại của Recast đã được tích hợp sẵn trong Unity gọi là NavMesh. Tuy nhiên Recast có rất nhiều thông số cấu hình, làm cho việc so sánh trở nên phức tạp hơn tại vì mỗi loại môi trường có thông số tối ưu khác nhau. Nhóm nghiên cứu thống nhất chọn một thông số cấu hình cân bằng giữa độ phức tạp của đồ thị trả về, thời gian chạy, độ chính xác đó là {Cell size = 5, Max Border Edge Length = 100} cho tất cả các môi trường.

Trước khi tiến hành đo thời gian xây dựng ECM, nhóm nghiên cứu đã tính trước số đỉnh hiện có của các vật cản (Obstacle vertices). Sau đó tính số đỉnh, số cạnh của ECM và Recast. Hình 6 hiển thị ECM khi được xây dựng xong. Hình 7 hiển thị Recast khi được xây dựng xong. Kết quả ta thu được ở bảng 1 và bảng 2.

Dựa vào bảng 1 và bảng 2, ta có thể thấy số đỉnh vật cản tăng lên làm thời gian xây dựng cũng tăng dần lên. Số đỉnh, số cạnh đồ thị của ECM nhiều hơn Recast ở các môi trường nhỏ và ít hơn ở các môi trường lớn. Số đỉnh, số cạnh biểu diễn độ phức tạp của đồ thị. Đồ thị có độ phức tạp ít hơn thì thời gian chạy giải thuật tìm đường nhanh hơn. Thời gian xây dựng của ECM nhanh hơn so với Recast ở tất cả các môi trường.

Bảng 1. Thời gian xây ECM.

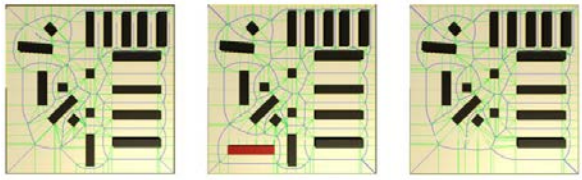
Môi trường	Số đỉnh vật cản	Số đỉnh đồ thị	Số cạnh đồ thị	Thời gian xây dựng (ms)
Military	84	181	394	37
City	286	566	1258	69
City 2x2	1144	2225	4946	278
City 4x4	4664	9042	20282	1222

Bảng 2. Thời gian xây Recast.

Môi trường	Số đỉnh vật cản	Số đỉnh đồ thị	Số cạnh đồ thị	Thời gian xây dựng (ms)
Military	84	141	308	145
City	286	338	742	158
City 2x2	1144	2584	5638	678
City 4x4	4664	10199	22854	2480

Thêm/xóa một vật cản: Để việc tính toán đơn giản hơn, nhóm nghiên cứu chỉ thêm/xóa

một vật cản có số đỉnh là 4. Ngoài ra, với trường hợp thêm một vật cản, ta chỉ thêm vào trong vùng không gian trống (không được chồng đè lên các vật cản đang có sẵn trong bản đồ). Minh họa trong hình 8. Kết quả ta tính toán trong bảng 3.



(a) Bản đồ ban đầu *(b) Thêm một vật cản* *(c) Xóa một vật cản*

Hình 8. *Thêm/xóa vật cản.*

Bảng 3. Thời gian tính toán khi thêm/xóa vật cản

Môi trường	Thời gian thêm (ms)	Thời gian xóa (ms)	Thời gian xây dựng ECM (ms)
Military	4	6.9	37
City	5.2	7.7	69
City 2x2	9.9	16	278
City 4x4	14.2	18.6	1222

Dựa vào bảng 3, ta nhận thấy thời gian tính toán khi xóa một vật cản có phần nhỉnh hơn so với thời gian thêm một vật cản. Nhưng nhìn chung, việc thêm hay xóa một vật cản đều được thực hiện rất nhanh, gần như không đáng kể so với việc xây dựng lại toàn bộ ECM, do giải thuật chỉ xét các ô ECM xung quanh đó.

5. Kết luận

Trong lĩnh vực Robotics, mô phỏng hay game, các đối tượng cần tính toán và di chuyển trong một không gian có thể di chuyển được. Lưới điều hướng là một cách tiếp cận phổ biến. Trong bài báo này nhóm nghiên cứu đã trình bày cách xây dựng bản đồ tường minh cho phép việc tìm đường có thể thực hiện trong thời gian thực, đồng thời ECM cũng phù hợp với tìm đường cho rô bốt và mô phỏng đám đông di chuyển.

ECM cũng hỗ trợ tốt các tác vụ như truy vấn các chướng ngại vật gần nhất, tính toán đường dẫn có độ mở (bán kính đi qua được) phù hợp. Đối với môi trường 2D có n vật cản, ECM cần $O(n)$ không gian và tính toán trong thời gian $O(n \log n)$.

Thực nghiệm cũng cho thấy việc tính toán chỉ tiêu tốn thời gian *ms* với môi trường có nhiều vật cản❑

Tài liệu tham khảo

[1] L.E. Kavraki, P. Švestka, J.-C. Latombe, and M.H. Overmars. *Probabilistic roadmaps for path planning in high-dimensional configuration spaces*. IEEE Transactions on Rô bốtics and Automation, 12(4):566–580, 1996;

[2] J.J. Kuffner and S.M. LaValle. *RRT-Connect: An efficient approach to single-query path planning*. Proceedings of the 17th IEEE International Conference on Rô bốtics and Automation, pages 995–1001, 2000;

[3] W.G. van Toll, A.F. Cook IV, and R. Geraerts. *Real-time density-based crowd simulation.* Computer Animation and Virtual Worlds, 23(1):59–69, 2012;

[4] G. Snook. *Simplified 3D movement and pathfinding using navigation meshes*. Mark DeLoura, editor, Game Programming Gems, pages 288–304. Charles River Media, 2000;

[5] P. Tozour. *Building a near-optimal navigation mesh*. Steve Rabin, editor, AI Game Programming Wisdom, pages 171–185. Charles River Media, 2002;

[6] S. Fortune. *A sweepline algorithm for Voronoi diagrams*. Algorithmica, 2:153–174, 1987;

[7] M.I. Shamos and D. Hoey. *Closest-point problems*. Proceedings of the 16th Annual IEEE Symposium on Foundations of Computer Science, pages 151–162, 1975;

[8] P.J. Green and R. Sibson. *Computing Dirichlet tessellations in the plane*. The Computer Journal, 21(2):168–173, 1978;

[9] K.E. Hoff III, T. Culver, J. Keyser, M. Lin, and D. Manocha. *Fast computation of generalized Voronoi diagrams using graphics hardware*. International Conference on Computer Graphics and Interactive Techniques, pages 277–286, 1999;

[10] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 3rd edition, 2008;

[11] Aron Granberg. *A* Pathfinding Project*. https://arongranberg.com/astar, 2019.

Ngày nhận bài: 23/8/2019
Ngày chuyển phản biện: 26/8/2019
Ngày hoàn thành sửa bài: 16/9/2019
Ngày chấp nhận đăng: 23/9/2019