ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

BÀI TẬP LỚN
ĐỀ TÀI: Ứng dụng fuzzy logic dể cân bằng xe

Sinh viên: Vũ Trung Đức - 21021577

Lớp: K66-E-EC2

Giảng viên phụ trách: TS. Nguyễn Thị Thanh Vân

Bộ môn: Hệ thống logic mờ

Lớp môn học: ELT3086_65

Hà Nội - 2024

MỤC LỤC

Giới thiệu đề tài

Trong bối cảnh phát triển công nghệ, việc ứng dụng các phương pháp trí tuệ nhân tạo vào điều khiển tự động ngày càng trở nên phổ biến. Một trong những lĩnh vực nổi bật là điều khiển xe tự hành, nơi mà việc ra quyết định nhanh chóng và chính xác là vô cùng quan trọng. Đề tài này tập trung vào việc mô phỏng ứng dụng logic fuzzy trong việc điều khiển xe, nhằm cải thiện khả năng phản ứng của xe với các tình huống khác nhau trên đường.

Logic fuzzy, với khả năng xử lý thông tin không chắc chắn và mơ hồ, là công cụ lý tưởng để giải quyết các vấn đề phức tạp trong điều khiển. Trong mô phỏng này, chúng tôi sử dụng hai đầu vào chính: khoảng cách từ xe đến vị trí giữa của đường và vận tốc của xe. Khoảng cách được định nghĩa trong khoảng từ -10 đến 10, trong khi vận tốc xe nằm trong khoảng từ 0 đến 100. Các đầu vào này sẽ được xử lý thông qua các quy tắc fuzzy để xác định đầu ra là góc xoay của xe, với giá trị nằm trong khoảng từ -90 đến 90 độ.

Mô phỏng được thực hiện trên nền tảng Unity 3D, cho phép tạo ra một môi trường trực quan và tương tác, giúp người dùng dễ dàng theo dõi và đánh giá hiệu quả của hệ thống điều khiển. Qua đó, đề tài không chỉ góp phần vào việc nâng cao hiểu biết về logic fuzzy mà còn mở ra hướng đi mới cho các ứng dụng trong lĩnh vực xe tự hành, hướng tới một tương lai an toàn và hiệu quả hơn trong giao thông.

Phân tích đề tài

Thiết kế chương trình

B1: Xác định giới hạn đầu vào

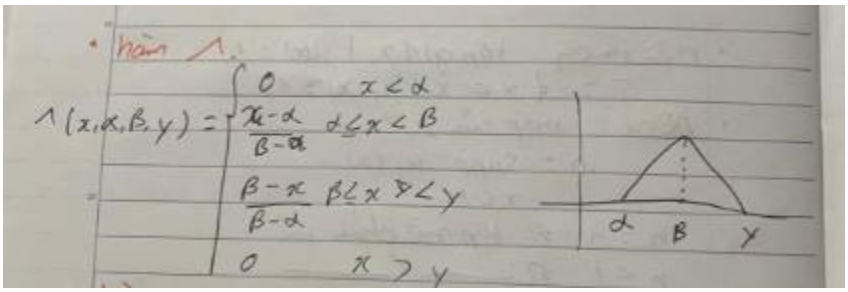
Đầu vào:

- Khoảng cách từ xe đến vị trí giữa của đường [-10, 10]
- Vận tốc xe: [0, 100]

Đầu ra: Góc xoay của xe [-90, 90]

B2: Mờ hóa

- Sử dụng hàm tam giác với các giá trị: anpha, beta, lamda

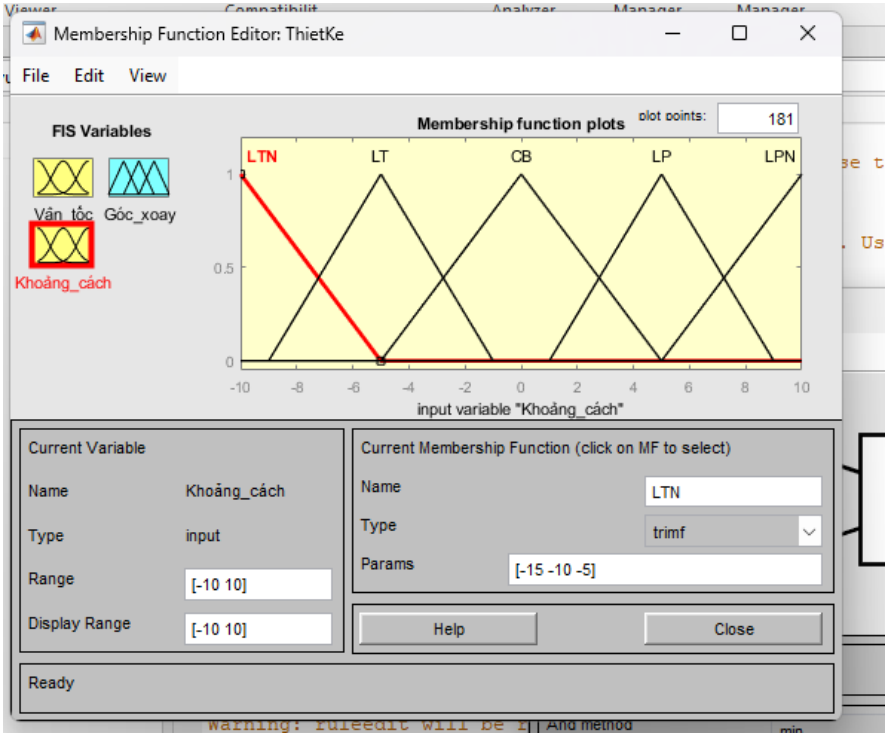


Hình định nghĩa hàm tam giác

- Vận tốc:
- Rất âm: tamGiac(-150, -100, -50)
- Âm: tamGiac(-100, -50, 0)
- Không: tamGiac(-50, 0, 50)
- Dương: tamGiac(0, 50, 100)
- Rất dương: tamGiac(50, 100, 150)

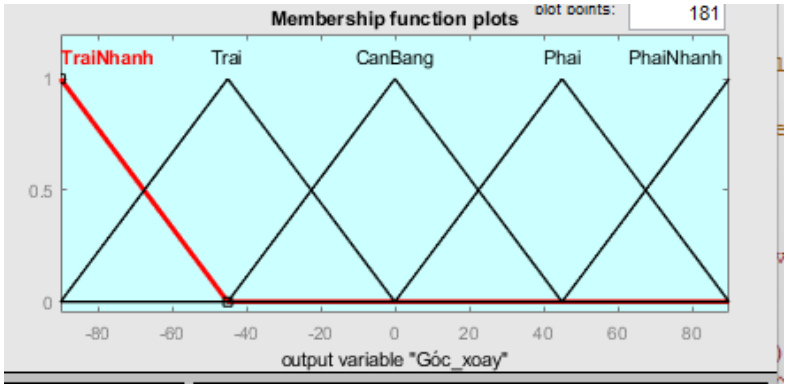
Hình định nghĩa đầu vào vận tốc trong mathlab

- Khoảng cách:
- Lệch trái nhiều: xe đang bị lệch về bên trái nhiều so với vị trí xét – tamGiac(-15, -10, -5)
- Lệch trái: xe đang bị lệch về bên trái so với vị trí xét – tamGiac(-9, -5, -1)
- Cân bằng: xe đang gần với vị trí xét – tamGiac(-5, 0, 5)
- Lệch phải: xe đang bị lệch về bên phải so với vị trí xét – tamGiac(1, 5, 9)
- Lệch phải nhiều: xe đang bị lệch về bên phải nhiều so với vị trí xét – tamGiac(5, 10, 15)



Hình định nghĩa đầu vào khoảng cách trong mathlab

- Góc xoay:
- Trái nhanh: góc xoay lớn về bên trái – tamGiac(-135, -90, -45)
- Trái: góc xoay về bên trái – tamGiac(-90, -45, 0)
- Cân bằng: góc xoay rất nhỏ – tamGiac(-45, 0, 45)
- Phải: góc xoay về bên phải – tamGiac(0, 45, 90)
- Phải nhanh: góc xoay lớn về bên phải – tamGiac(45, 90, 135)



Hình định nghĩa đầu ra góc trong mathlab

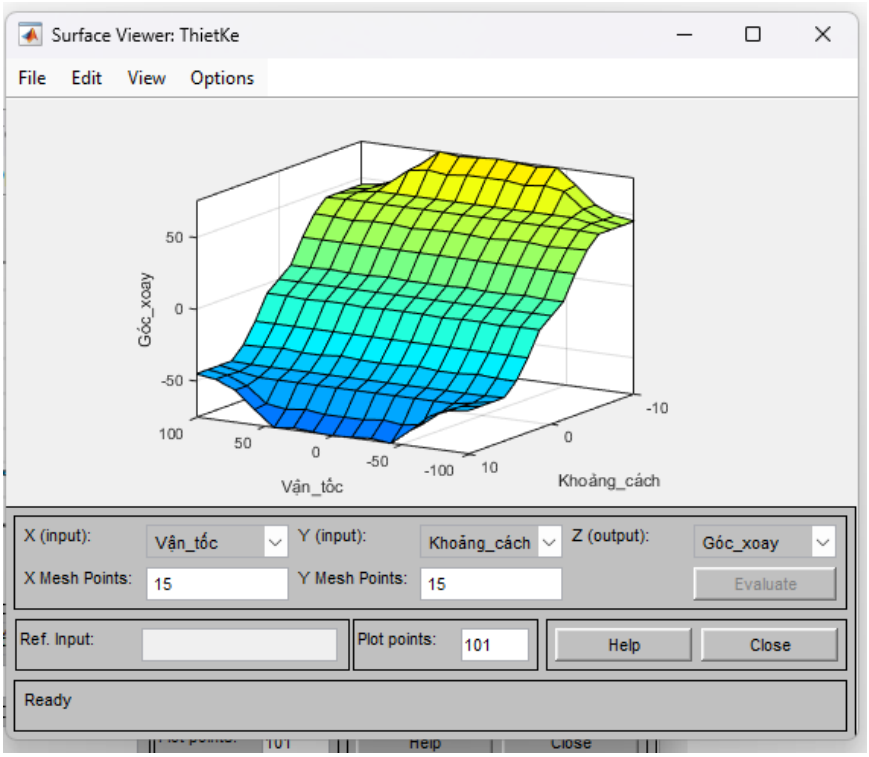
B3: Xác định luật:

- Nếu Xe đi rất nhanh: rất âm/dương
    - Nếu lệch trái nhiều => thay đổi góc sang phải nhỏ
    - Nếu lệch phải nhiều => thay đổi góc sang trái nhỏ
    - Nếu lệch trái => thay đổi góc sang phải nhỏ
    - Nếu lệch phải => thay đổi góc sang trái nhỏ
    - Nếu cân bằng => để góc ở trạng thái cân bằng
- Nếu Xe đi chậm: âm/dương/không
    - Nếu lệch trái nhiều => thay đổi góc sang phải lớn
    - Nếu lệch phải nhiều => thay đổi góc sang trái lớn
    - Nếu lệch trái => thay đổi góc sang phải nhỏ
    - Nếu lệch phải => thay đổi góc sang trái nhỏ
    - Nếu cân bằng => để góc ở trạng thái cân bằng

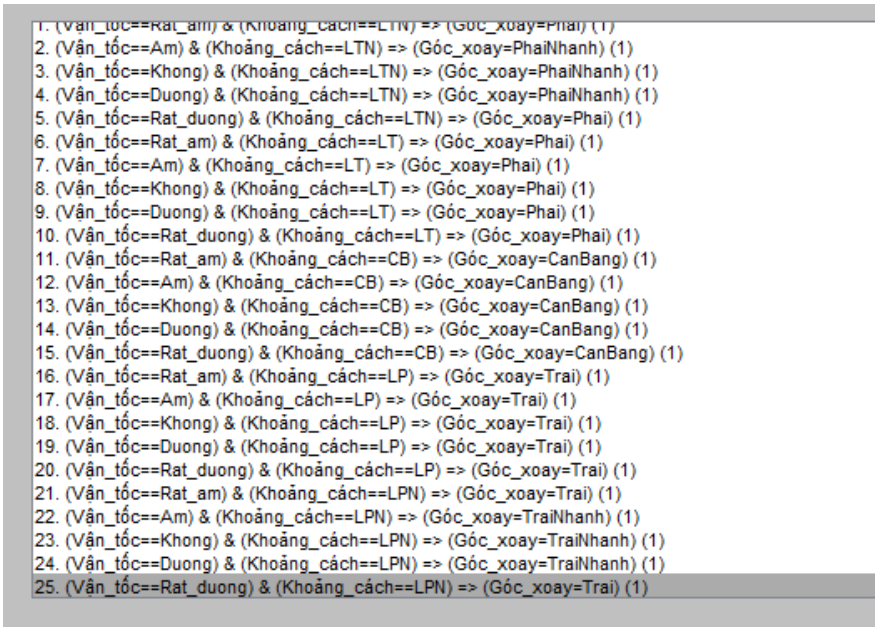
Ta có bảng:

	LTN	LT	CB	LP	LPN
Rat am	Phai	Phai	CB	Trai	Trai
Am	Phai Nhanh	Phai	CB	Trai	Trai Nhanh
Khong	Phai Nhanh	Phai	CB	Trai	Trai Nhanh

Duong	Phai Nhanh	Phai	CB	Trai	Trai Nhanh
Rat duong	Phai	Phai	CB	Trai	Trai



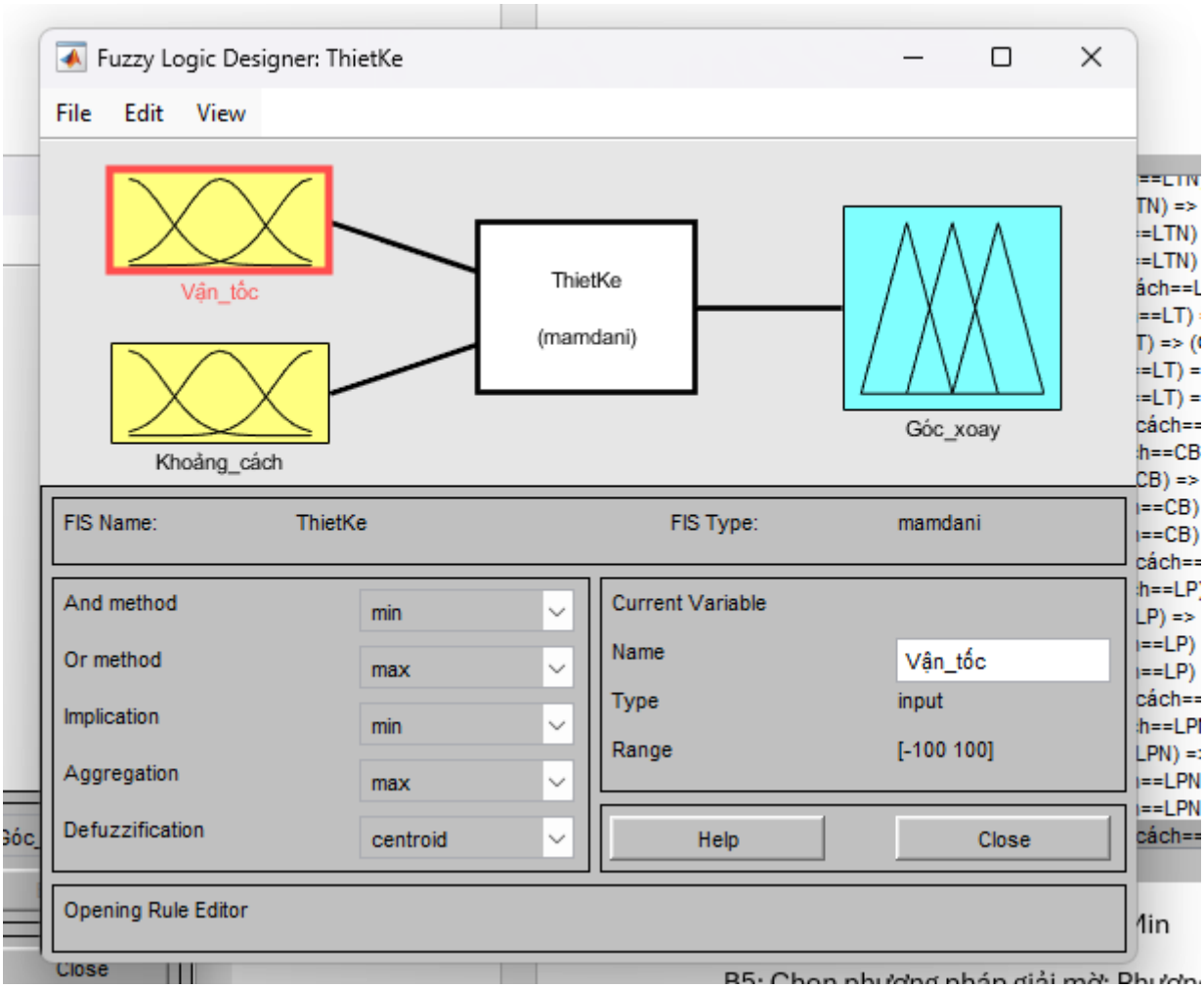
Hình ảnh hiển thị thông tin trong mathlab



Hình ảnh thêm các luật trên mathlab

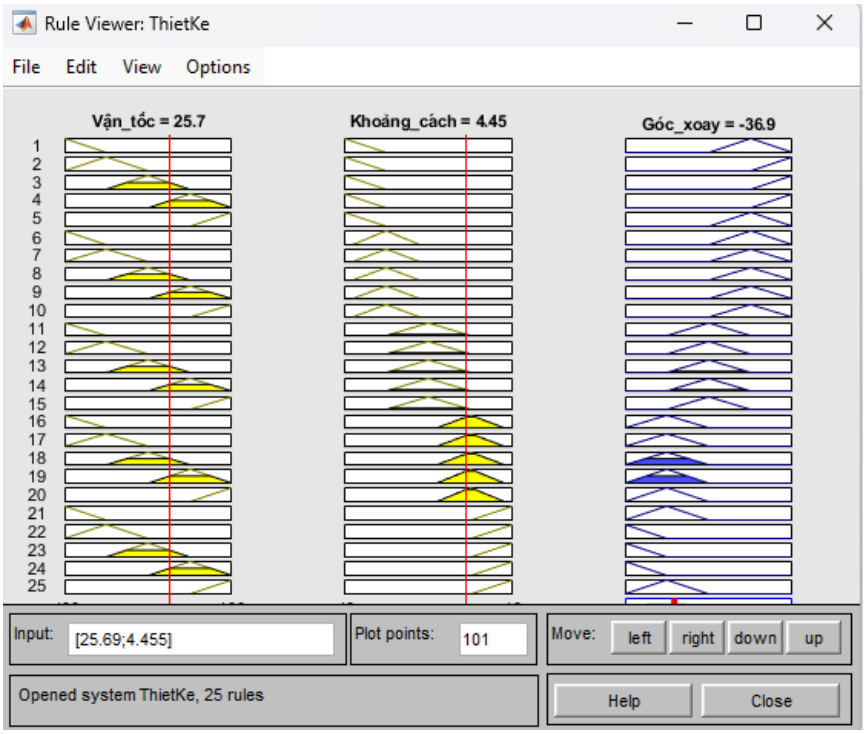
B4: Chọn cơ chế suy diễn: Max – Min

B5: Chọn phương pháp giải mờ: Phương pháp điểm trọng tâm



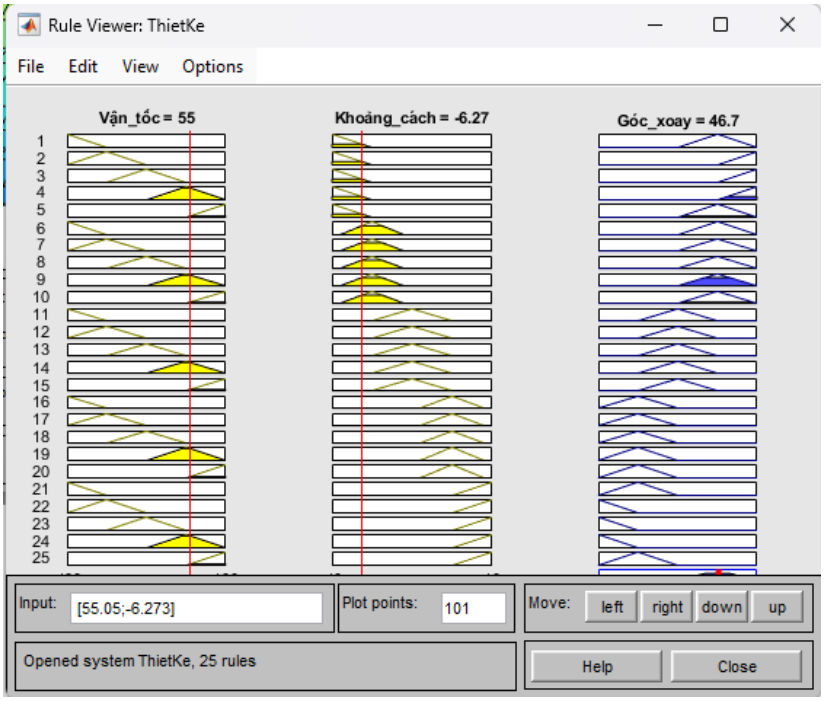
Hình ảnh cài đặt các thông số tính toán trong mathlab

B6: Kiểm tra đánh giá:



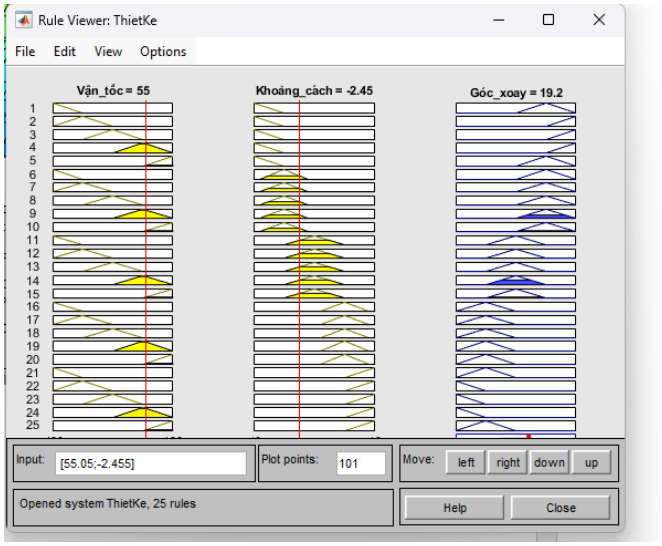
Hình ảnh khi vận tốc vừa, khoảng cách lệnh phải, góc xoay sang trái

- Khi vận tốc vừa và xe bị lệch bên phải => xe cần xoay sang trái để tiến tới vị trí cân bằng



Hình ảnh khi vận tốc vừa, khoảng cách lệch trái nhiều, xe quay sang phải nhanh

- Khi vận tốc vừa mà khoảng cách lệch phải nhiều => xe cần quay sang phải nhanh để tiến tới vị trí cân bằng



Hình ảnh khi vận tốc vừa, khoảng cách lệch trái ít, xe quay sang phải

- Khi vận tốc vừa mà khoảng cách lệch phải it => xe cần quay sang phải để tiến tới vị trí cân bằng

Thực hiện mô phỏng trên Unity

- B1: Định nghĩa các biến ngôn ngữ

```
listKc.Add(new BienNgonNgu("LTN", GetTriangle(valueStartKc, valueEndKc, -15, -10, -5)));
listKc.Add(new BienNgonNgu("LT", GetTriangle(valueStartKc, valueEndKc, -9, -5, -1)));
listKc.Add(new BienNgonNgu("CB", GetTriangle(valueStartKc, valueEndKc, -5, 0, 5)));
listKc.Add(new BienNgonNgu("LP", GetTriangle(valueStartKc, valueEndKc, 1, 5, 9)));
listKc.Add(new BienNgonNgu("LPN", GetTriangle(valueStartKc, valueEndKc, 5, 10, 15)));

listVt.Add(new BienNgonNgu("RatAm", GetTriangle(valueStartVt, valueEndVt, -150, -100, -50)));
listVt.Add(new BienNgonNgu("Am", GetTriangle(valueStartVt, valueEndVt, -100, -50, 0)));
listVt.Add(new BienNgonNgu("Khong", GetTriangle(valueStartVt, valueEndVt, -50, 0, 50)));
listVt.Add(new BienNgonNgu("Duong", GetTriangle(valueStartVt, valueEndVt, 0, 50, 100)));
listVt.Add(new BienNgonNgu("RatDuong", GetTriangle(valueStartVt, valueEndVt, 50, 100,
150)));

listAngle.Add(new BienNgonNgu("TraiNhanh", GetTriangle(valueStartAngle, valueEndAngle, -
135, -90, -45)));
listAngle.Add(new BienNgonNgu("Trai", GetTriangle(valueStartAngle, valueEndAngle, -90, -45,
0)));
listAngle.Add(new BienNgonNgu("CanBang", GetTriangle(valueStartAngle, valueEndAngle, -45,
0, 45)));
listAngle.Add(new BienNgonNgu("Phai", GetTriangle(valueStartAngle, valueEndAngle, 0, 45,
90)));
listAngle.Add(new BienNgonNgu("PhaiNhanh", GetTriangle(valueStartAngle, valueEndAngle, 45,
90, 135)));
```

- B2: Thực hiện tính toán:

- Tính khoảng cách từ xe tự điều khiển với xe người dùng

```
vDistance = (transAI.transform.position.x - transPlayer.transform.position.x)*2;
```

- Lấy vận tốc hiện tại của xe người dùng

```
vSpeed = sliderSpeed.value*10;
```

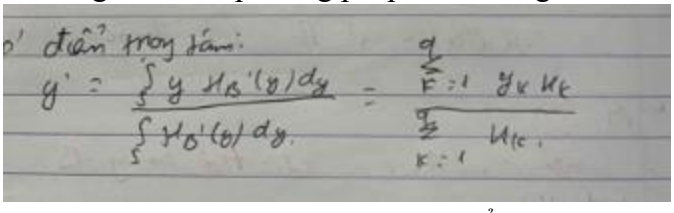
- Lấy mảng các giá trị nguy tương ứng của đầu vào khoảng cách
- Lấy mảng các giá trị nguy tương ứng của đầu vào vận tốc

```
nguyKc = GetNguy(listKc, vDistance);
nguyVt = GetNguy(listVt, vSpeed);
```

- Thực hiện tính giá trị nguy của biến các biến ngôn ngữ góc (Max – Min) từ các luật định nghĩa trước đó:

```
float traiNhanh = Mathf.Max(Mathf.Min(nguyKc[4], nguyVt[1]),
                            Mathf.Min(nguyKc[4], nguyVt[2]),
                            Mathf.Min(nguyKc[4], nguyVt[3]));

float trai = Mathf.Max(Mathf.Min(nguyKc[3], nguyVt[0]),
                       Mathf.Min(nguyKc[3], nguyVt[1]),
                       Mathf.Min(nguyKc[3], nguyVt[2]),
                       Mathf.Min(nguyKc[3], nguyVt[3]),
                       Mathf.Min(nguyKc[3], nguyVt[4]),
                       Mathf.Min(nguyKc[4], nguyVt[0]),
                       Mathf.Min(nguyKc[4], nguyVt[4]));

float canBang = Mathf.Max(Mathf.Min(nguyKc[2], nguyVt[0]),
                       Mathf.Min(nguyKc[2], nguyVt[1]),
                       Mathf.Min(nguyKc[2], nguyVt[2]),
                       Mathf.Min(nguyKc[2], nguyVt[3]),
                       Mathf.Min(nguyKc[2], nguyVt[4]));

float phai = Mathf.Max(Mathf.Min(nguyKc[1], nguyVt[0]),
                       Mathf.Min(nguyKc[1], nguyVt[1]),
                       Mathf.Min(nguyKc[1], nguyVt[2]),
                       Mathf.Min(nguyKc[1], nguyVt[3]),
                       Mathf.Min(nguyKc[1], nguyVt[4]),
                       Mathf.Min(nguyKc[0], nguyVt[0]),
                       Mathf.Min(nguyKc[0], nguyVt[4]));

float phaiNhanh = Mathf.Max(Mathf.Min(nguyKc[0], nguyVt[1]),
                            Mathf.Min(nguyKc[0], nguyVt[2]),
                            Mathf.Min(nguyKc[0], nguyVt[3]));
```

- Thực hiện tính giá trị trọng tâm theo phương pháp điểm trọng tâm:



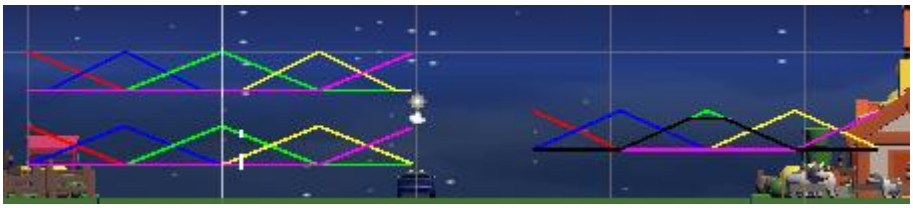
Hình ảnh công thức phương pháp điểm trọng tâm

```
float tuSo = 0;
float mauSo = 0;
List<ToHop> result = new List<ToHop>();
for (int i = 0; i < n; i++)
{
    float tn = Mathf.Min(listAngle[0].listValue[i].nguy, traiNhanh);
    float t = Mathf.Min(listAngle[1].listValue[i].nguy, trai);
    float cb = Mathf.Min(listAngle[2].listValue[i].nguy, canBang);
    float p = Mathf.Min(listAngle[3].listValue[i].nguy, phai);
    float pn = Mathf.Min(listAngle[4].listValue[i].nguy, phaiNhanh);
    result.Add(new ToHop(listAngle[0].listValue[i].x, Mathf.Max(tn, t, cb, p, pn)));
    tuSo += result[i].nguy * result[i].x;
    mauSo += result[i].nguy;
}
float angle = tuSo / mauSo;
```

- Hiển thị thông số lên màn hình:

```
    transAI.transform.localRotation = Quaternion.Euler(0, angle, 0);
    DrawTriangles(result, colorLine[5], pStartDrawAngle, doPhongDaiAngle);
    txtAngle.text = "Góc xoay: " + angle.ToString("F2") + " độ";
    txtKc.text = "Khoảng cách: " + vDistance.ToString("F2") + " m";
    txtV.text = "Vận tốc: " + (vSpeed).ToString("F2") + " m/s";
```

Kết quả đạt được



Hình ảnh biểu diễn đầu vào, đầu ra và đường bao khi tính toán

Hình ảnh khi vận tốc vừa, xe không lệch, góc xoay ở vị trí cân bằng

Hình ảnh xe di chuyển vừa, lệch phải nhiều, xe quay trái nhanh

Hình ảnh xe di chuyển nhanh, lệch phải vừa, xe quay phải vừa

Link video demo: Fuzzy logic in Unity3d controll car

Hướng phát triển

Dự án "Ứng dụng Fuzzy Logic để Cân Bằng Xe" đã đạt được mục tiêu ban đầu nhưng có thể mở rộng thêm các chức năng để cải thiện sản phẩm:

1. **Tích hợp Cảm Biến Thực Tế**:
    - Kết hợp các cảm biến như lidar, camera, và cảm biến siêu âm để thu thập dữ liệu thực tế về môi trường xung quanh. Điều này sẽ giúp hệ thống xử lý thông tin chính xác hơn và ra quyết định tốt hơn trong các tình huống thực tế.
2. **Mở Rộng Quy Tắc Fuzzy**:
    - Phát triển thêm các quy tắc fuzzy phức tạp hơn để xử lý nhiều biến đầu vào cùng lúc, như địa hình, điều kiện thời tiết và trạng thái giao thông.
3. **Cải Thiện Giao Diện Người Dùng**:
    - Tạo ra một giao diện người dùng trực quan hơn, cho phép người dùng tương tác và điều chỉnh các tham số điều khiển. Điều này có thể giúp người dùng dễ dàng tùy chỉnh hệ thống theo nhu cầu riêng.
4. **Học Máy và Tự Học**:
    - Kết hợp các phương pháp học máy để cho phép hệ thống tự học từ các tình huống thực tế và cải thiện độ chính xác qua thời gian. Điều này có thể bao gồm việc sử dụng mạng nơ-ron để tối ưu hóa các quy tắc fuzzy.
5. **Mô Phỏng Các Tình Huống Khác Nhau**:
    - Mở rộng mô phỏng để bao gồm các tình huống giao thông phức tạp hơn, chẳng hạn như điều kiện đường xá, người đi bộ và các phương tiện khác. Điều này sẽ giúp cải thiện khả năng ứng phó của hệ thống.
6. **Tích Hợp Hệ Thống Điều Khiển Động Lực**:
    - Nghiên cứu và phát triển các thuật toán điều khiển động lực học cho xe, cho phép hệ thống điều khiển không chỉ góc xoay mà còn cả vận tốc và gia tốc của xe.
7. **Kiểm Tra và Đánh Giá Thực Tế**:
    - Triển khai các thử nghiệm thực tế để đánh giá hiệu quả của hệ thống trong môi trường thực. Điều này sẽ cung cấp những thông tin quý báu để điều chỉnh và cải thiện hệ thống.

Những hướng phát triển này không chỉ giúp nâng cao khả năng của dự án mà còn tạo ra những cơ hội mới cho việc áp dụng logic fuzzy trong các lĩnh vực khác nhau của công nghệ điều khiển tự động.

Kết luận

Dự án "Ứng dụng Fuzzy Logic để Cân Bằng Xe" đã thành công trong việc xây dựng một hệ thống điều khiển xe tự hành dựa trên logic fuzzy, với mục tiêu cải thiện khả năng phản ứng của xe trong các tình huống khác nhau. Qua quá trình nghiên cứu và phát triển, em đã áp dụng thành công các quy tắc fuzzy để xử lý hai đầu vào chính: khoảng cách từ xe đến vị trí giữa của đường và vận tốc xe.

Sử dụng Unity 3D, mô phỏng đã được thực hiện một cách trực quan, cho phép người dùng dễ dàng theo dõi và đánh giá hiệu quả của hệ thống. Hệ thống đã sử dụng các hàm tam giác để mờ hóa dữ liệu đầu vào, từ đó xác định góc xoay của xe với độ chính xác cao. Kết quả của mô phỏng cho thấy khả năng điều khiển xe một cách mềm mại và linh hoạt, giúp xe duy trì vị trí cân bằng ngay cả khi gặp phải các tình huống không chắc chắn.

Bên cạnh đó, dự án cũng đã mở ra nhiều hướng nghiên cứu mới trong lĩnh vực điều khiển xe tự hành. Việc áp dụng logic fuzzy cho phép xử lý các yếu tố không chắc chắn trong môi trường thực tế, góp phần tạo ra các giải pháp an toàn và hiệu quả hơn. Em hy vọng rằng những kết quả đạt được từ dự án sẽ có thể được áp dụng rộng rãi trong các ứng dụng thực tế, từ xe tự hành đến các hệ thống điều khiển tự động khác.

Cuối cùng, dự án không chỉ giúp nâng cao kiến thức về logic fuzzy mà còn củng cố kỹ năng lập trình và phát triển phần mềm trong môi trường Unity 3D. Em tin rằng những kiến thức và kinh nghiệm thu được sẽ là nền tảng vững chắc cho những nghiên cứu tiếp theo trong lĩnh vực này.

Tài liệu tham khảo

[1] **Zadeh, L. A.** (1965). Fuzzy Sets. *Information and Control*, 8(3), 338-353.
DOI: 10.1016/S0019-9958(65)90241-X

[2] **Mamdani, E. H., & Assilian, S.** (1975). An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller. *International Journal of Man-Machine Studies*, 7(1), 1-13.
DOI: 10.1016/S0020-7373(75)80002-2

[3] **Ross, T. J.** (2010). Fuzzy Logic with Engineering Applications. *Wiley*.
ISBN: 978-0-470-51373-1

[4] **Hollerbach, J. M.** (1993). Fuzzy Logic Control for Autonomous Vehicles. In *Proceedings of the IEEE International Conference on Robotics and Automation* (pp. 104-109).
DOI: 10.1109/ROBOT.1993.291024

[5] **Pugh, S.** (1991). Total Design: Integrated Methods for Successful Product Engineering. *Addison-Wesley*.
ISBN: 978-0-201-50350-1

[6] **Unity Technologies.** (2023). *Unity User Manual*. Retrieved from https://docs.unity3d.com/Manual/index.html