SIMULAÇÃO DE SISTEMA AUTÔNOMO PARA TESTES VEICULARES

Thiago Mendes Pereira[1], Lúcia Valéria de Ramos Arruda[2]

[1]Robert Bosch Limitada, UTFPR-CPGEI
[2]UTFPR-CPGEI

E-mails: thiago.pereira@br.bosch.com, lvrarruda@utfpr.edu.br

RESUMO

A indústria automotiva possui um severo processo de validação para a liberação de um veículo para produção seriada. Um dos passos mais importantes nesse processo de liberação são os testes de durabilidade. Veículos são dirigidos em uma pista controlada por longos períodos de tempo sem interrupção. Em alguns casos, as pistas simulam ambientes rigorosos, como irregularidades no pavimento da estrada. Essas condições severas podem ser desconfortáveis ou até mesmo insalubres para os motoristas de testes. Este trabalho descreve o desenvolvimento de um robô em ambiente de simulação, capaz de contornar essas condições de teste, utilizando sensores, atuadores e um software de controle baseado em Robotic Operating System (ROS).

INTRODUÇÃO

Os grandes fabricantes de veículos possuem em suas fábricas pistas de testes como a mostrada na figura 1, para validação dos modelos que serão liberados para produção em série, a fim de validar alterações nos produtos, testar novas funcionalidades ou até mesmo para realizar uma auditoria de qualidade dos veículos produzidos em massa. Essas pistas possuem obstáculos que simulam condições de dirigibilidade reais, onde vários componentes podem ser testados, dependendo do ciclo escolhido. Por exemplo, imperfeições no asfalto como as mostradas na figura 2, são frequentemente utilizadas para teste de suspensão. Além disto, obstáculos são utilizados para testes do sistema de direção e o motor é testado em áreas de aceleração, entre outras condições de teste.

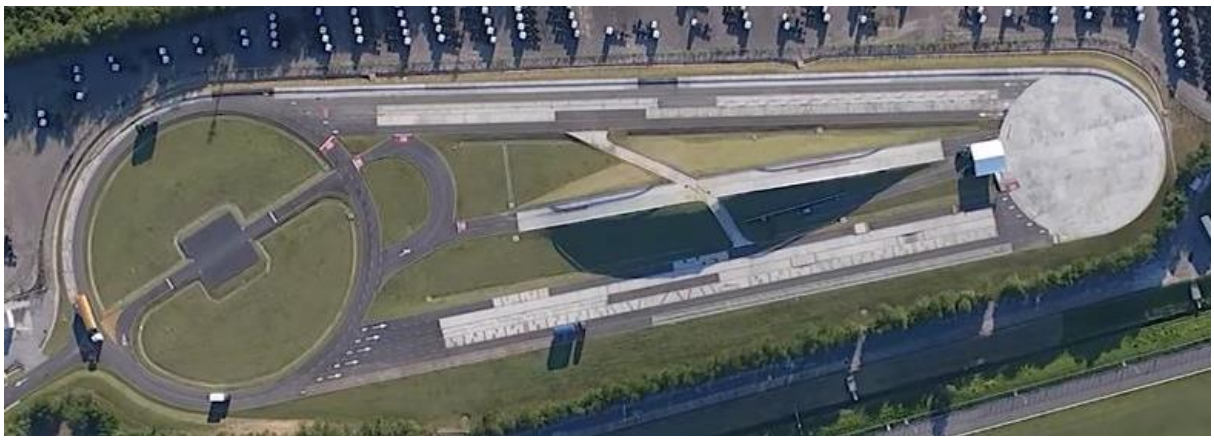

Figura 1 – pista de testes de montadora de caminhões

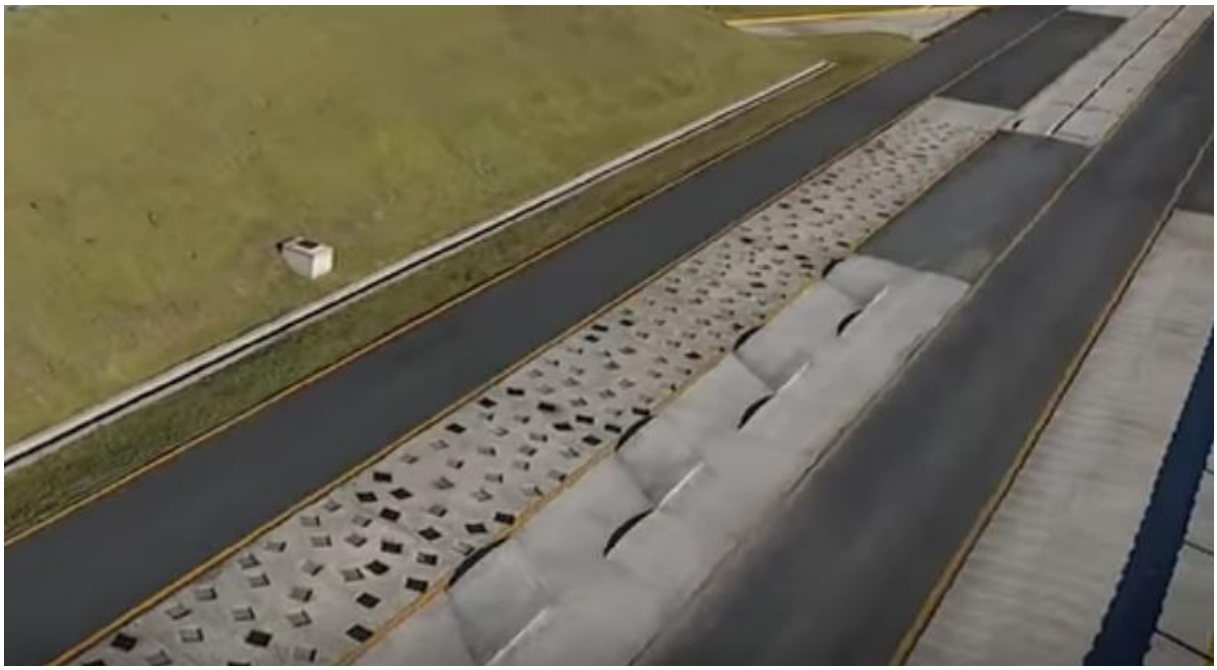

Figura 2 – Irregularidades no asfalto para testes de suspensão

A duração dos testes veiculares é normalmente bastante longa e ininterrupta, o que gera desconforto para os motoristas. Em algumas rotinas de teste de suspensão, por exemplo, a oscilação da cabine do veículo é tão intensa e repetitiva, que tal teste é considerado insalubre. Em adição, o fator humano também pode introduzir variabilidade no teste dificultando a consolidação de seus resultados, uma vez que cada motorista possui um estilo de condução diferente, mesmo seguindo rotinas padrão.

Para atenuar tais problemas, propõe-se um simulador computacional dotado de um "condutor robótico", capaz de reproduzir o ciclo de testes em uma pista fechada, simulando algumas das condições pelas quais o veículo é submetido em uma rotina de validação. O sistema precisa ser programado com uma rota estabelecida, identificar a sua localização atual, controlar a aceleração, desaceleração e direção do veículo na rota, sendo capaz de reagir a obstáculos aleatórios que podem surgir durante o percurso, como animais ou pessoas.

A ferramenta de simulação para automação de teste veicular proposta neste artigo é baseada em sistemas de controle fuzzy (Pasino e Yurkovich, 1998), e é desenvolvida na plataforma de simulação V-REP (Rohmer et al., 2013), utilizando o sistema operacional ROS: Robot Operating System, (Quigley et al., 2009).

1. DESCRIÇÃO DO SISTEMA

1.1 Estrutura do sistema

O ambiente de simulação para desenvolvimento do ambiente de teste automotivo foi implementado em VREP. O VREP é um simulador de robótica onde é possível simular computacionalmente, por meios de modelos cinemáticos e dinâmicos, o comportamento de um robô em ambiente virtual, descrevendo sem que seja necessário efetivamente construir o robô (Rohmer et al., 2013). Assim, o modelo do veículo e da pista de testes foram construídos no VREP.

A cena do VREP se comunica com um nó ROS responsável por todo o controle do movimento do veículo. ROS, como o próprio nome sugere, é uma coleção de frameworks de software para desenvolvimento de robôs, que fornece a funcionalidade de um sistema operacional em um cluster de computadores heterogêneo.
Os algoritmos de controle e os procedimentos de teste foram desenvolvidos na plataforma ROS, que é também responsável pela comunicação e gerenciamento dos diversos algoritmos compondo o simulador desenvolvido. A figura 3 traz a arquitetura do sistema, mostrando a estruturação dos componentes, bem como a interação entre eles através de mensagens. Os procedimentos para planejamento da rota, controle de velocidade e a direção são implementados em ROS, utilizando algumas das bibliotecas disponíveis. O módulo de controle (figura 3) é ainda responsável por analisar a distância dos obstáculos, informada pelo veículo, e tomar a decisão de qual ação será tomada.

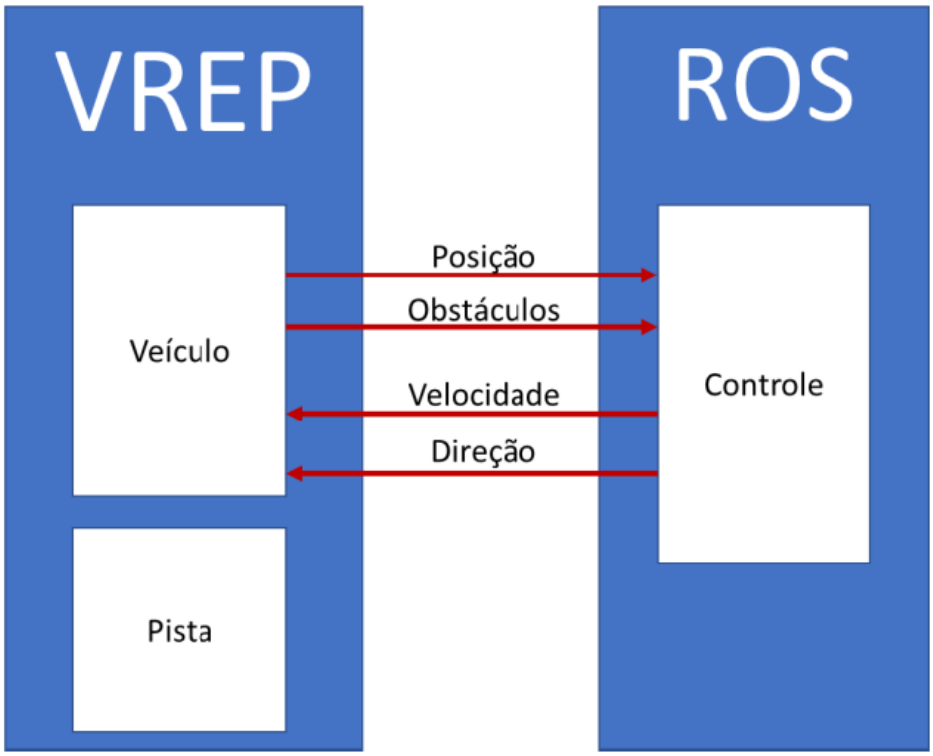

Figura 3 – arquitetura do sistema

1.2. Pista de testes

A pista de testes é um circuito fechado semioval, inspirado em pistas de montadoras de veículos. Ela possui aproximadamente 500 metros de comprimento por 12 metros de largura, conforme ilustrado na figura 4. A pista possui obstáculos para reproduzir condições de testes. Um deles simula direção em slalom para teste do sistema de direção. Foram utilizadas barreiras perpendiculares ao sentido do movimento do veículo para que ele seja forçado a sair da trajetória planejada, mostrando a capacidade de desvio de obstáculos. Paralelamente ao sentido do movimento foram também adicionadas barreiras nas bordas da pista para que o veículo não escolha uma rota de desvio por fora da área asfaltada, como mostrado na figura 5.

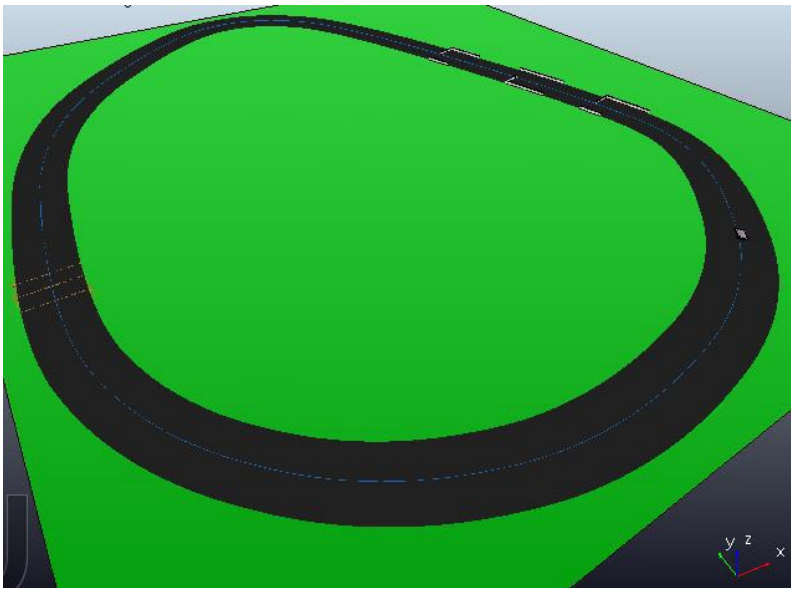

Figura 4 – modelagem da pista de testes

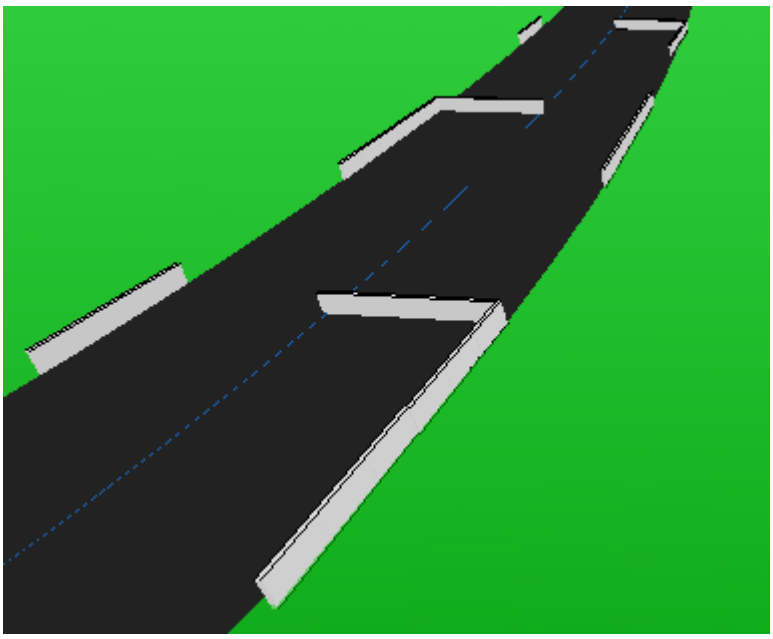

Figura 5 – modelagem dos obstáculos para manobra

Também foi contemplado na pista, uma área com irregularidades no asfalto para teste da suspensão do veículo (Figura 6). O sistema autônomo deve ser capaz de passar por elas sem oscilação e desestabilização na trajetória. Finalmente, foi previsto a aparição inesperada de pessoas ou outros objetos dinâmicos no meio da pista.

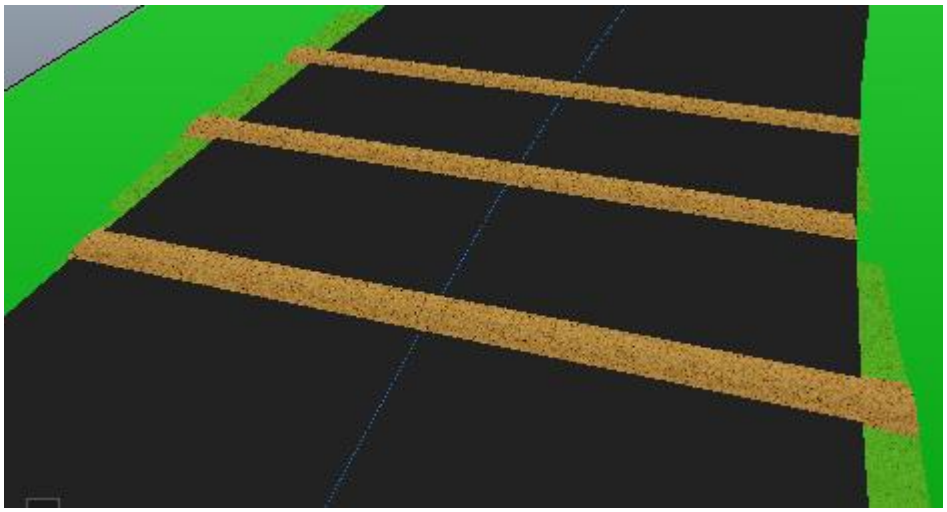

Figura 6 – modelagem das irregularidades na pista

1.3. Veículo

O veículo é uma representação simplificada da geometria de Ackermann, onde as rodas dianteiras tracionam e se movem de maneira que o centro do raio da curva está sempre na projeção do eixo traseiro, conforme esquematizado na figura 7. A modelagem também inclui o sistema mola/amortecedor das suspensões do veículo.

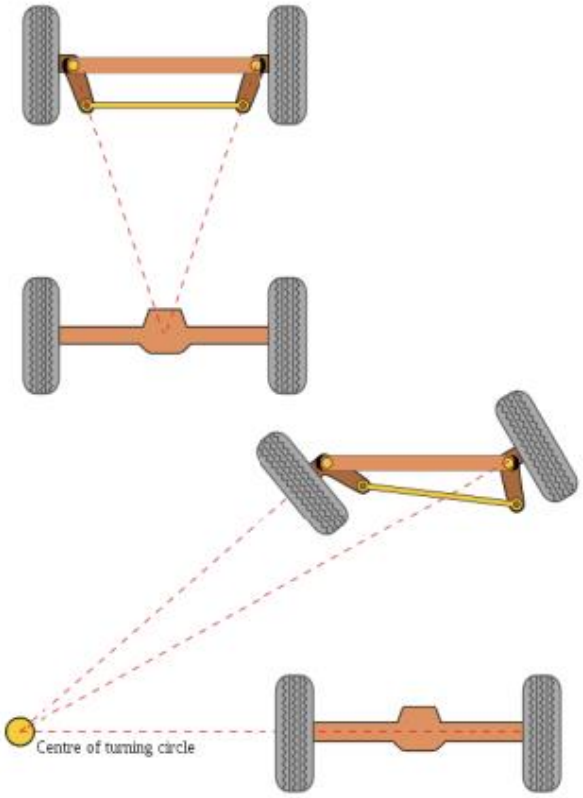

Figura 7 – A geometria de Ackermann

Na dianteira do veículo, está instalado um sensor Laser Hokuyo URG-04LX-UG01 (figura 8) que emite um feixe de laser, capaz de detectar objetos até 10 metros de distância em um range angular de 180º. Este feixe de laser, ao colidir com um objeto, reflete para o sensor. Sabida a velocidade de propagação da luz pelo ar, o sensor calcula a distância do objeto através do tempo entre a emissão e a recepção do pulso luminoso. O sensor retorna para o software de controle um vetor de 512 posições, contendo a distância para os objetos onde o laser refletiu. Cada posição do vetor representa uma distância em relação a um ângulo que varia de 0 a 180º (steps de 0,35º a cada posição do vetor). Esta informação é utilizada pela estrutura de controle para a tomada de decisões com relação à atuação no freio e sistema de direção.

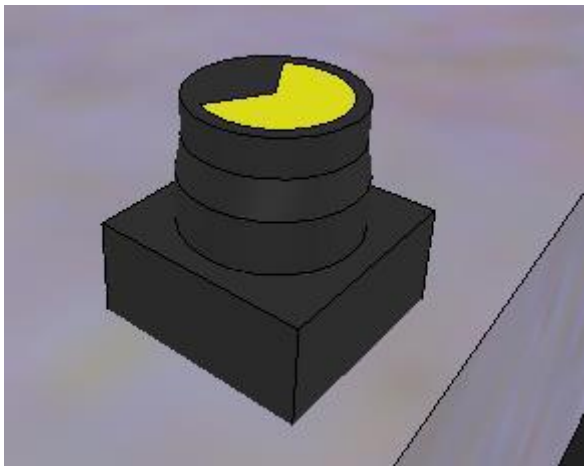

Figura 8 – modelagem do sensor laser

2. SOFTWARE DE CONTROLE

O sistema de controle desenvolvido é dividido em 2 módulos, sendo um deles executado no VREP e outro no ROS. O módulo de controle ROS recebe do módulo VREP as leituras de odometria do veículo e a medição do sensor laser e devolve para VREP o comando de velocidade para o automóvel. Este fluxo de informação é mostrado na figura 9.

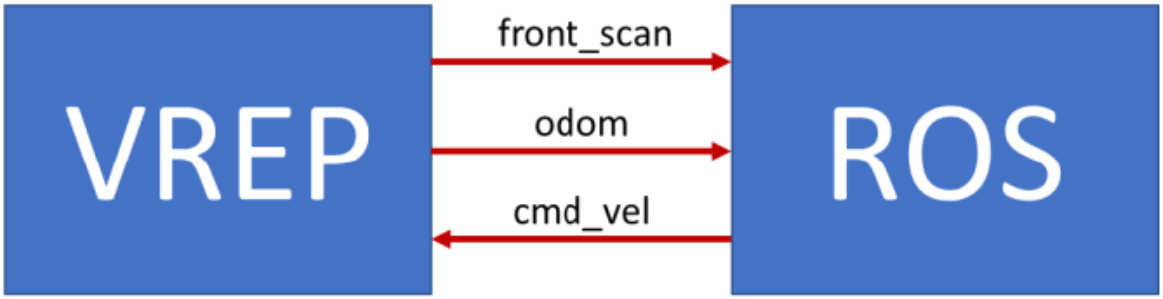

Figura 9 – Fluxo de informação entre os Módulos de controle

2.1. Módulo de controle no VREP

O módulo de controle executado no VREP constantemente subscreve no tópico "*cmd_vel*", oriundo do nó de controle ROS. Neste tópico estão contidas as instruções de comando de aceleração e frenagem do veículo (Twist/linear.x) e o comando da direção do veículo (Twist/angular.z). Estas mensagens são utilizadas pelo VREP para simular a movimentação

do veículo pela cena. Por outro lado, este módulo publica periodicamente no tópico odom a posição no plano cartesiano (mensagens Odometry/pose.position.x e Odometry/pose.position.y) em que o veículo se encontra, simulando um GPS RTK de alta precisão.
Também publica periodicamente a distância dos objetos detectados pelo sensor Laser. Estes dados são enviados para o controlador ROS pelo tópico front_scan, através de um vetor de 512 posições, contendo as distâncias para os objetos, com steps de 0,35º (de 0º a 180º). Esse vetor é armazenado na mensagem LaserScan/ranges.

2.2. Módulo de controle ROS

O módulo de controle ROS é o centro de tomada de decisões do sistema. Ele é responsável pelas seguintes funções: planejamento da rota, seguimento de rota e desvio de obstáculos. Cada uma dessas funcionalidades é apresentada a seguir.

2.2.1. Planejamento da rota

A pista é dividida em 11 segmentos, conforme a Figura 10. Cada vértice dos segmentos é um ponto de referência para o planejamento de rota.

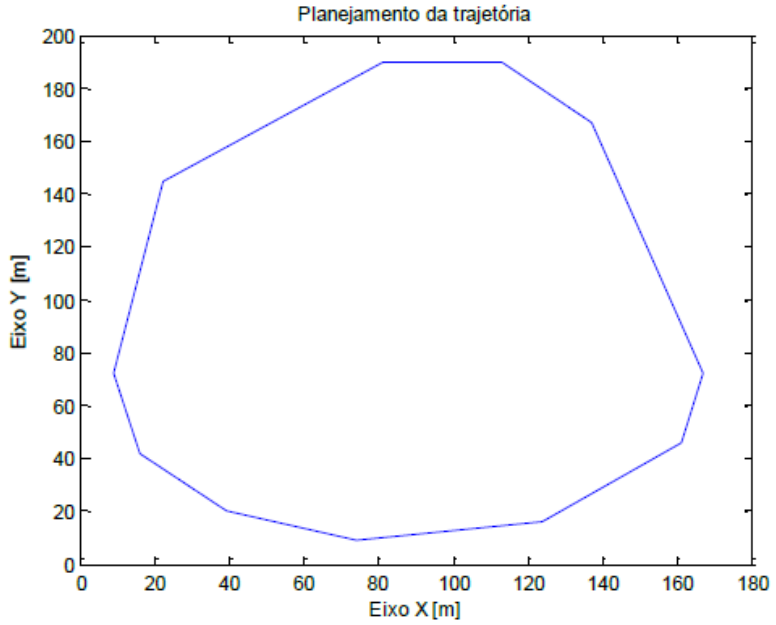

Figura 10 – vértices e segmentos da rota

O planejamento de rota interpola uma trajetória nova entre 2 vértices através do algoritmo Quintic, cada vez que o veículo chega em um novo vértice.
O algoritmo Quintic determina uma trajetória que pode ser descrita através de um polinômio de 5ª ordem.

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$$

Derivando a função, obtém-se o polinômio de velocidade e derivando mais uma vez, o polinômio de aceleração.

$$q(t_0) = a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3 + a_4 t_0^4 + a_5 t_0^5$$

$$v(t_0) = a_1 + 2a_2 t_0 + 3a_3 t_0^2 + 4a_4 t_0^3 + 5a_5 t_0^4$$

$$\alpha(t_0) = 2a_2 + 6a_3 t_0 + 12a_4 t_0^2 + 20a_5 t_0^3$$

$$q(t_f) = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 + a_4 t_f^4 + a_5 t_f^5$$

$$v(t_f) = a_1 + 2a_2 t_f + 3a_3 t_f^2 + 4a_4 t_f^3 + 5a_5 t_f^4$$

$$\alpha(t_f) = 2a_2 + 6a_3 t_f + 12a_4 t_f^2 + 20a_5 t_f^3$$

Colocando os polinômios na forma matricial, obtém-se:

$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 \\ 0 & 1 & 2t_0 & 3t_0^2 & 4t_0^2 & 5t_0^4 \\ 0 & 0 & 2 & 6t_0 & 12t_0^2 & 20t_0^3 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^2 & 5t_f^4 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} q_0 \\ v_0 \\ \alpha_0 \\ q_f \\ v_f \\ \alpha_f \end{bmatrix}$$

Ao se estabelecer os valores de posição iniciais e finais, caracterizados pelos vértices da rota pode-se definir os coeficientes do polinômio que descreve a trajetória e, consequentemente, o caminho que o veículo percorrerá no deslocamento entre 2 vértices adjacentes. Nesta simulação, cada deslocamento entre 2 vértices possui 5 pontos intermediários planejados pelo algoritmo Quintic.

2.2.2. Seguimento da rota

Uma vez que os pontos da trajetória foram definidos pela função Quintic, cabe ao controlador de rota fazer com que o veículo siga o planejamento.

O ângulo da direção é controlado através de um controlador PID conforme malha de realimentação da figura 11. O algoritmo constantemente calcula o erro para o controlador através da diferença entre o ângulo da trajetória (ângulo formado pelo vetor entre a posição atual do veículo e o próximo ponto planejado na rota) e o ângulo de guinada (yaw) do veículo. A saída deste controlador é limitada em +/- 45º para que o movimento da geometria de Ackermann seja plausível em relação a um veículo real.

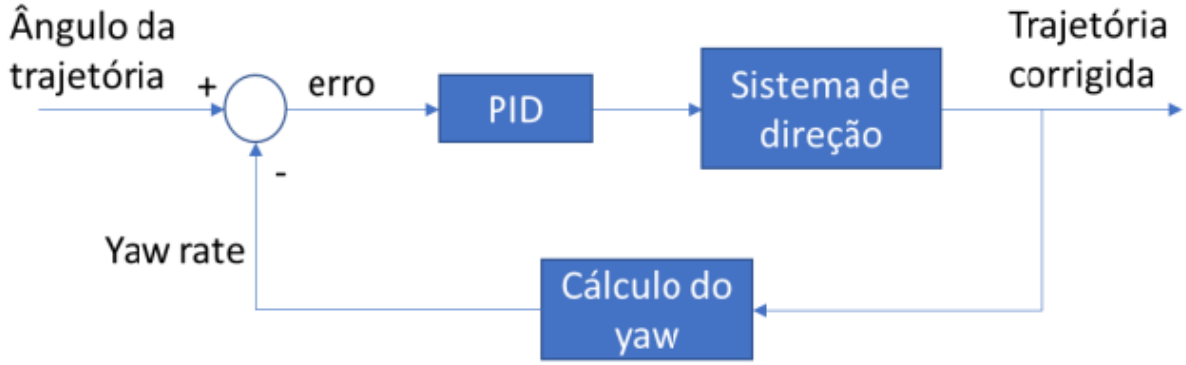

Figura 11 – controlador PID de ângulo de direção

O ciclo de testes é organizado por faixa de velocidades de acordo com o elemento que se quer avaliar (obstáculos na pista ou irregularidades). A velocidade na trajetória é dividida em 3 áreas constantes. Na 1ª área, que vai do segmento 1 ao 5, a velocidade é mantida em 8 m/s (aprox. 29 km/h). Neste trecho, a presença de obstáculos pode fazer com que a velocidade seja reduzida, devido à atuação do controlador de desvio de obstáculos discutido a seguir. Na 2ª área, que compreende o segmento 6, a velocidade é reduzida para 5 m/s (18 km/h) para que o veículo passe pelas irregularidades do asfalto. Na 3ª área, que vai do segmento 7 até o 11, o veículo desenvolve a velocidade máxima permitida na pista de 11,11 m/s (40 km/h), para fechar o ciclo de testes.

2.2.3. Desvio de obstáculos

O desvio de obstáculos é realizado através de um controlador *fuzzy*. Este controlador foi implementado no código-fonte através da utilização da biblioteca FuzzyLite do ROS. As entradas do controlador Fuzzy são os dados informados pelo sensor laser e as saídas são atuações no freio e na direção do veículo (figura 12).

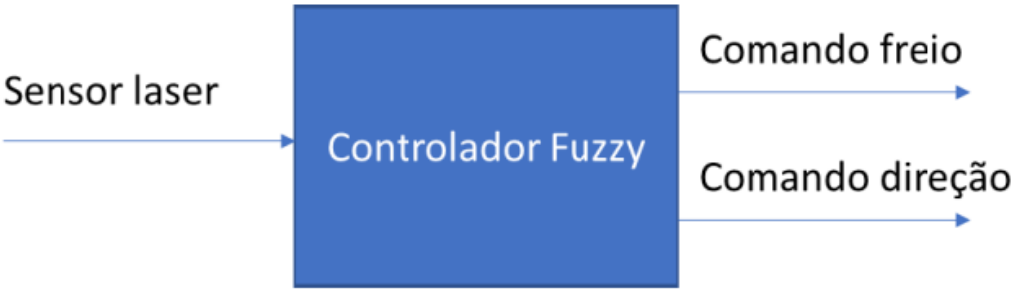

Figura 12 – Esquemático do controlador Fuzzy

A interpretação dos dados oriundos do sensor laser envolve a divisão do feixe de 180º (vetor com as distâncias de 512 posições, de 0,35º cada, fornecido pelo sensor Hokuyo) em 5 segmentos de, aproximadamente 102º, trabalhando como 5 sensores independentes, chamados de SensorMuitoEsquerda, SensorEsquerda, SensorMeio, SensorDireita, SensorMuitoDireita, e esquematizado na figura 13. Os dados utilizados pelo controlador fuzzy são as menores distâncias localizadas por cada um destes 5 segmentos do sensor. Desta maneira, o controlador pode saber se o obstáculo está logo à frente, à sua direita ou à sua esquerda.

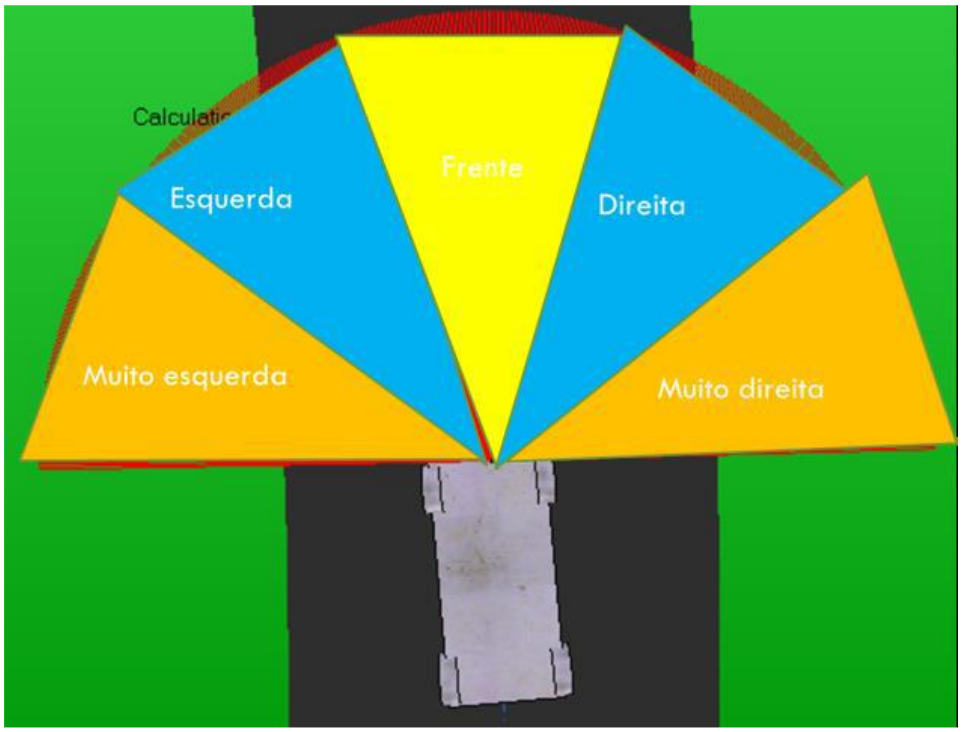

Figura 13 – Divisão de segmentos do sensor laser

As distâncias em cada um dos 5 sensores também são consideradas pelo controlador *fuzzy*. Quando nenhum objeto é detectado, o controlador compreende a distância como *ForaRange*. Caso o objeto detectado esteja até 2 metros de distância do veículo, ele é considerado perto. Se estiver por volta de 6 metros, é considerado médio, se estiver por volta de 10 metros, é considerado longe. As funções de pertinência associadas a estes predicados são dadas na figura 14.

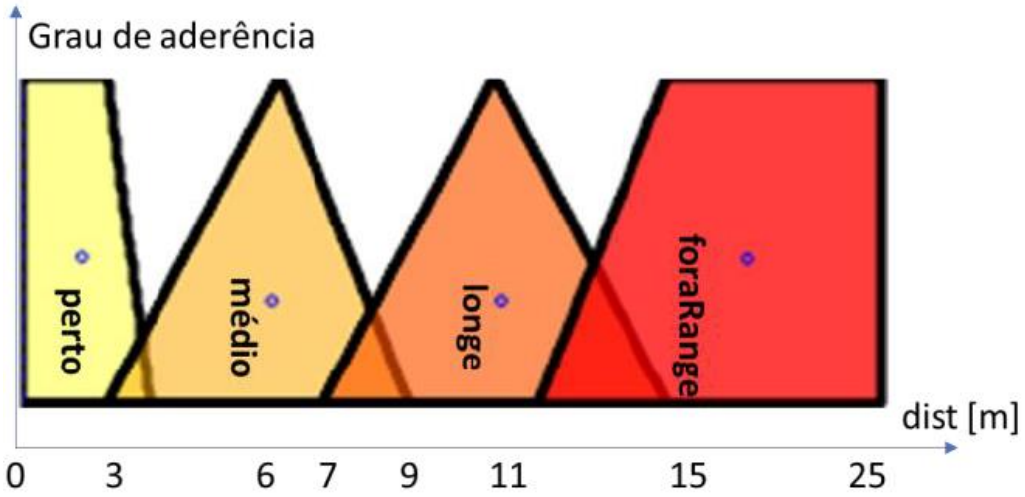

Figura 14 – Funções de pertinência de entrada

De acordo com os valores fuzzy da leitura dos 5 sensores laser, o controlador fuzzy calcula a taxa de freio e o ângulo da direção do veículo. A base de regras para o freio representa a seguinte análise: quanto mais próximo o veículo se encontra de um obstáculo, mais ele freia em relação à velocidade que se deslocava antes do controle atuar. Caso ele se aproxime demais de um obstáculo que não consegue desviar, o veículo para completamente. A pressão

de freio (variável de saída) é classificada em 4 diferentes níveis (freiaNada, freiaPouco, freiaMedio, freiaMuito) conforme as funções de pertinências dadas na figura 15.

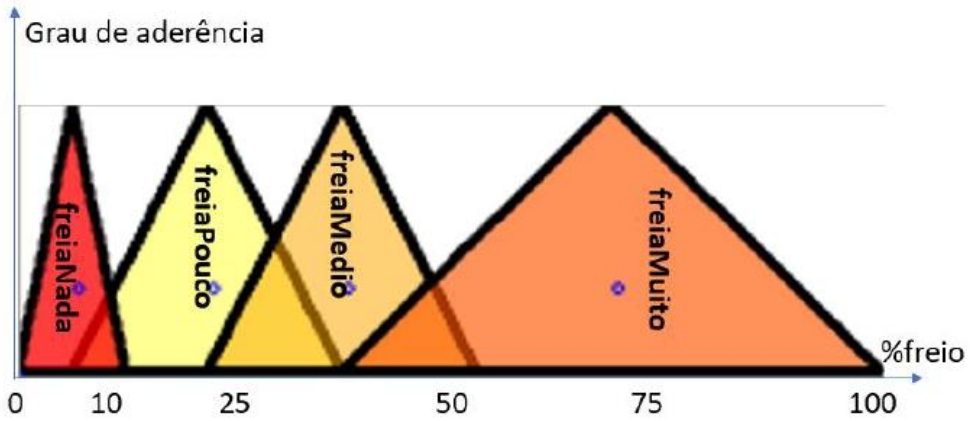

Figura 15 – Funções de pertinência de saída: acionamento do freio

Algumas regras intuitivas para controle do freio são:
• Se todos os 5 sensores informarem que não há obstáculos no range de alcance, não atue no freio (freiaNada).
• Se algum dos 5 sensores detectar um objeto em uma distância classificada como longe, atue pouco no freio (freiaPouco).
• Se algum dos 5 sensores detectar um objeto em uma distância classificada como média, atue médio no freio (freiaMedio).
• Se algum dos 5 sensores detectar um objeto em uma distância classificada como perto, atue muito no freio (freiaMuito).
O controle da direção funciona de forma semelhante, sendo o ângulo da direção comandado pelas distâncias detectadas em cada um dos sensores laser. Os predicados utilizados na atuação na direção são classificados como viraMuitoEsquerda, viraEsquerda, viraReto, viraDireita, viraMuitoDireita, conforme figura 16.

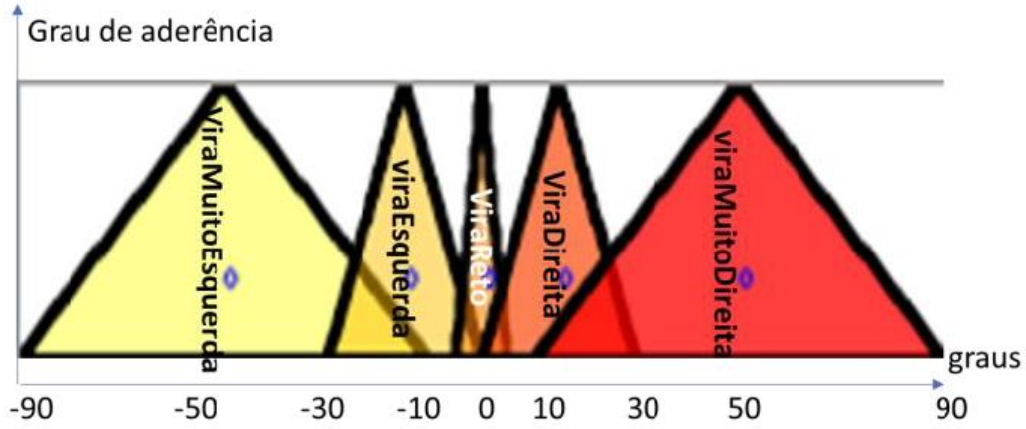

Figura 16 – Funções de pertinência de saída: Graus de atuação no ângulo de direção

Algumas regras que comandam a direção são:
• Se os sensores MuitoEsquerda ou MuitoDireita estiverem fora do Range ou longe, continue indo reto.

• Se os sensores MuitoEsquerda ou MuitoDireita estiverem a uma distância média, vire um pouco para a direção contrária ao obstáculo.
• Se os sensores MuitoEsquerda ou MuitoDireita estiverem a uma distância perto, vire muito para a direção contrária ao obstáculo.
• Se os sensores Direita ou Esquerda estiverem fora do Range, continue indo reto.
• Se os sensores Direita ou Esquerda estiverem a uma distância longe ou média, vire pouco para a direção contrária ao obstáculo.
• Se os sensores Direita ou Esquerda estiverem a uma distância perto, vire muito para a direção contrária ao obstáculo.
Existem ainda um controlador de emergência para desvio reativo, caso uma pessoa ou obstáculo móvel adentre a pista repentinamente. Se ela estiver imediatamente à frente do veículo, o carro para. Existem ainda outras regras, responsáveis pelo desvio do veículo em presença de pessoas que reproduzem as seguintes situações:
• Se somente o sensor do meio detectar um obstáculo e os demais não, isso significa que há uma pessoa (ou algum obstáculo não padrão) no caminho. Enquanto estiver longe, desvie pouco para a esquerda para sair do caminho do obstáculo. (base de regras para ângulo de direção)
• Caso o desvio não seja suficiente e a pessoa se encontre a uma distância média ou perto, freie muito para evitar uma colisão. (base de regras para freio)
A combinação de todas as regras mencionadas anteriormente forma a lógica de controle de desvio de obstáculos do módulo de controle ROS.
Enquanto o controlador fuzzy não atua, o veículo é controlado pelo PID seguidor de rota. No momento em que uma intervenção fuzzy é necessária, devido à presença de um obstáculo, o controlador PID é desligado e o fuzzy assume o comando, pois possui maior prioridade. Após o completo do desvio do obstáculo, a intervenção Fuzzy não é mais necessária, e o PID volta a atuar.

RESULTADOS

Após vários testes, o veículo foi capaz de percorrer o percurso conforme o planejado, desviando de barreiras fixas (figura 17) e obstáculos aleatórios (figura 18), percorrendo um asfalto irregular (figura 19) e mantendo a velocidade e direção planejadas em cerca de 5 minutos (tempo de simulação). No entanto, devido à velocidade de processamento, a simulação dura cerca de 25 minutos para rodar em uma máquina virtual Linux embarcada em um PC Intel Core I5-7200U, clock 2.5GHz, 8GB de RAM, placa gráfica integrada Intel HD Graphics 620.

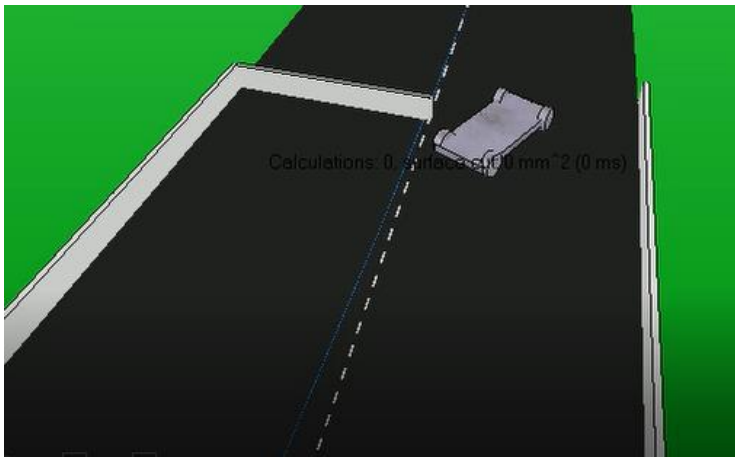

Figura 17 – Veículo desviando de obstáculos

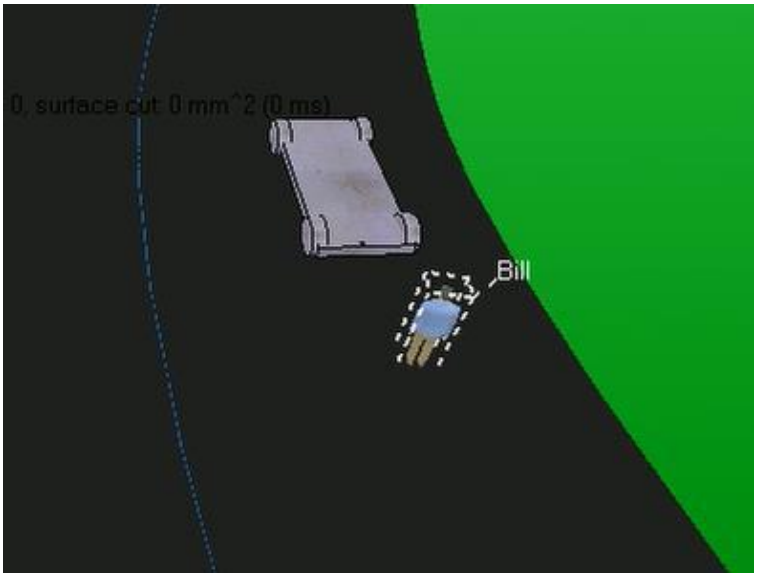

Figura 18 – Veículo parando perante a um pedestre

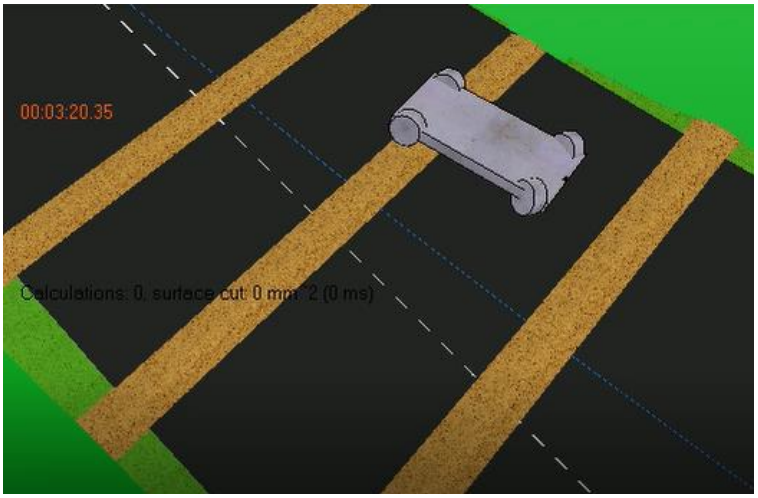

Figura 19 – Veículo passando por irregularidades na pista

CONCLUSÃO

A técnica de planejamento de rota por algoritmo Quintic, com seguidor PID e controlador de desvios Fuzzy se mostrou eficiente para executar a função de automação dos testes veiculares na cena proposta, reagindo a possíveis invasões aleatórias à pista.
Como passos de aprimoramento da solução, sugere-se uma calibração mais fina do controlador fuzzy, possibilitando uma variação mais suave da direção. Também pode-se evoluir no sentido de se adicionar mais um sensor (por exemplo, uma câmera) para diferenciação da área asfaltada e da grama. Assim, o veículo se mantém na pista e desvia de obstáculos sem a necessidade de se adicionar barreiras longitudinais nas bordas da pista.

REFERÊNCIAS

[1] E. Rohmer, S. P. Singh, and M. Freese, **"V-REP: A versatile and scalable robot simulation framework"**, in Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on. IEEE, 2013, pp. 1321–1326.

[2] Ferreira, T., Garcia, O., Vaqueiro J., **"Software Architecture for an autonomous car simulation using ROS, Morse and a QT based software for control and monitoring"**, SBAI - Simpósio Brasileiro de Automação Inteligente, 2015.

[3] Llorca, D. F., Milanes, V., Alonso, I. P., Gavillán, M., Daza, I. G., Pérez, J., Sotelo, M. A., **"Autonomous Pedestrian Collision Avoidance Using a Fuzzy Steering Controller"**, IEEE Transactions on Intelligent Transportation System, Vol. 12, nº 2, June 2011.

[4] Passino, K. Yurkovich, S. **"Fuzzy Control"**, Addison-Wesley, California, USA, 1998.

[5] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, **"ROS: an open-source Robot OperatingSystem"**, in ICRA workshop on open source software, vol. 3, no. 3.2. Kobe, 2009, p. 5.