DESENVOLVIMENTO DE UM SISTEMA DE MAPEAMENTO E LOCALIZAÇÃO PARA VEÍCULOS AUTÔNOMOS DO TIPO FORMULA STUDENT GERMANY UTILIZANDO O ALGORITMO FASTSLAM 1.0

Palavras-Chave: MAPEAMENTO, LOCALIZAÇÃO, FILTRO DE KALMAN, FILTRO DE PARTÍCULAS

Autores:
GUILHERME GRENDEL – FEM, UNICAMP
Prof. Dr. RODRIGO MOREIRA BACURAU (orientador) – FEM, UNICAMP

INTRODUÇÃO:

Na categoria de carros autônomos da competição *Formula Student Germany* os veículos competidores devem ser capazes de percorrer uma pista delimitada por cones no menor tempo possível. A fim de fazê-lo, é necessário estimar qual é a rota mais eficiente a ser percorrida e também qual deve ser o ângulo do volante e o estado do freio e acelerador para executá-la. Para isso, é essencial um mapa preciso dessa pista assim como uma boa estimativa da *pose* (posição e orientação) desse carro com relação a ela.

Por isso, algoritmos de *Simultaneous Localization And Mapping* (SLAM) são de grande importância em tais competições: eles não são apenas de capazes de estimar a posições dos cones como também de utilizar essa estimativa para aferir a *pose* do veículo, além de realizarem fusão de dados. Eles possibilitam, assim, a integração de sensores como GPS, *encoders* e IMU's, tipicamente utilizados para determinar localização, com câmeras e LIDAR's a fim de criar um mapa da pista em tempo real.

Muitos algoritmos de SLAM já foram desenvolvidos para a competição *Formula Student Germany* desde que a categoria para veículos autônomos foi lançada [1], incluindo variações dos algoritmos EKF SLAM, FastSLAM e GraphSLAM [2,3,4,5,6]. De modo geral, o GraphSLAM apresenta resultados superiores, porém, possui elevado custo computacional, o que pode tornar o FastSLAM e EKF SLAM mais adequados para sistemas computacionais de menor potência [6].

SIMULTANEOUS LOCALIZATION AND MAPPING (SLAM):

Seguindo o que foi apresentado na sessão anterior, este projeto propõe utilizar um algoritmo de SLAM para navegação autônima de um veículo de competição *Formula Student Germany*. Algoritmos de SLAM são, em sua maioria, filtros Bayesianos, e por isso costumam contar com etapas de predição e correção. Nesses filtros, o estado a ser estimado é frequentemente constituído pelas *poses* do veículo ou robô de um instante de tempo inicial até um instante *t*, e um mapa constituído pelos pontos onde estão localizadas as *landmarks*, que no contexto deste projeto são os cones.

Na implementação proposta neste trabalho, adota-se a suposição de que a localização dos cones não se altera com o tempo. Assim sendo, a etapa de predição reduz-se à estimativa da *pose* mais recente do carro a partir da estimativa da *pose* anterior e da estimativa do deslocamento entre os dois (odometria). Esse deslocamento pode ser calculado a partir da integração dos valores de aceleração fornecidos por uma IMU, integração dos valores de velocidade linear obtidos a partir de *encoders* nas rodas, e do deslocamento angular do volante, também obtido com um *encoder*. Já a etapa de correção será executada sobre todos os parâmetros de estado usando leituras de câmera.

FASTSLAM 1.0:

A família de algoritmos de SLAM escolhida para este projeto foi o FastSLAM 1.0 [7]. Ela se caracteriza pela divisão da função distribuição de probabilidade (f.d.p.) que representa a estimativa de estado em várias outras f.d.p: uma para a *pose* e uma para cada *landmark*. Para a primeira, utiliza-se um filtro de partículas e para as restantes utiliza-se filtros de Kalman estendidos. Cada partícula é composta pelos parâmetros da *pose* e pelos vetores média (μ) e matriz de covariância (Σ) de cada *landmark*, conforme ilustrado na Figura 1.

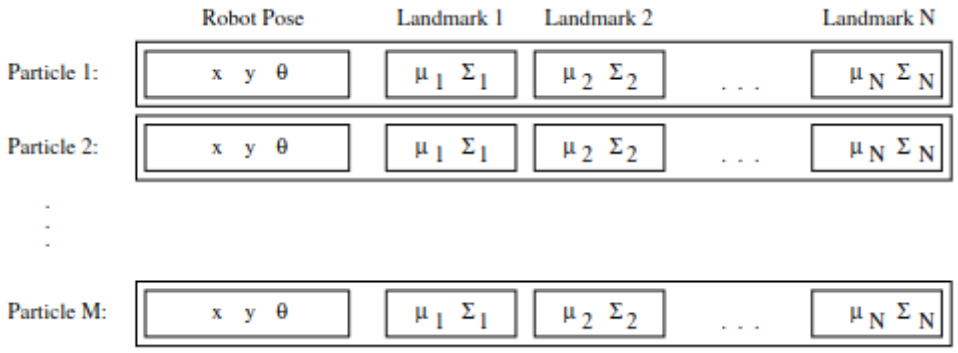

Figura 1 - Partículas do FastSLAM. Retirado de [7]

O *FastSLAM 1.0* é dividido em três etapas independentes: *sampling*, *map update* e *resampling*. Na etapa de *sampling*, novas *poses* são sorteadas aletatóriamente de acordo com uma função distribuição de probabilidade arbitrária $p(s_t | s_{t-1}, u_t)$, onde s_t é a *pose* num instante *t* e u_t a entrada de controle num instante de tempo *t*. Essa função distribuição de probabilidade poderia, por exemplo, ser gaussiana com média (*pose* da partícula) dada por uma função determinística e matriz de covariância ajustada a partir de testes.

Na etapa de *map update* as estimativas de localização das *landmarks* são corrigidas utilizando um filtro de Kalman estendido. Para isso se suporá que a localização delas não se altera com o tempo e serão utilizadas medições de LIDAR ou câmera.

Por último, na etapa de *resampling* é feita a atribuição de um peso para cada partícula. Elas então são sorteadas com reposição de acordo com esse peso. De maneira geral, as partículas cujas estimativas de localização das *landmarks* são mais condizentes com as medições de LIDAR ou câmera terão peso maior e portanto terão maior chance de "sobreviverem".

Na Figura 2 são representadas as etapas 1 e 3 do FastSLAM *1.0*. Em (a) são exibidas as partículas sorteadas e em (b) são destacadas as que restaram da etapa de *resampling*.

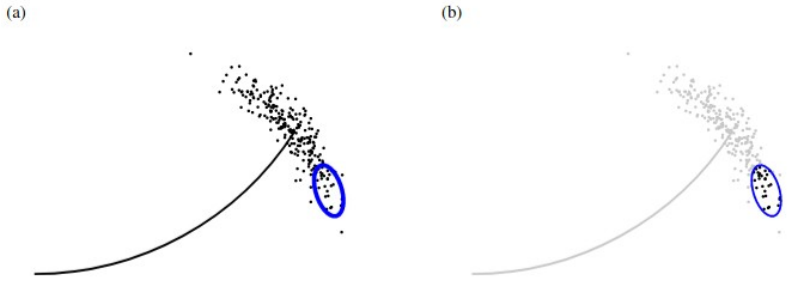

Figura 2 - Representação gráfica das etapas 1 e 3 do FastSLAM 1.0. Retirado de [7].

METODOLOGIA:

Neste trabalho, o algoritmo de FastSLAM 1.0 foi implementado em linguagem C. Essa linguagem foi escolhida porque o algoritmo base utiliza paradigma de programação procedural, programas escritos em C tendem a ter boa performance porque são compilados ao invés de interpretados e porque C é uma das linguagens de programação mais utilizadas em sistemas embarcados. Além disso, tanto o autor quanto o orientador já possuíam experiência com a linguagem antes do início do projeto.

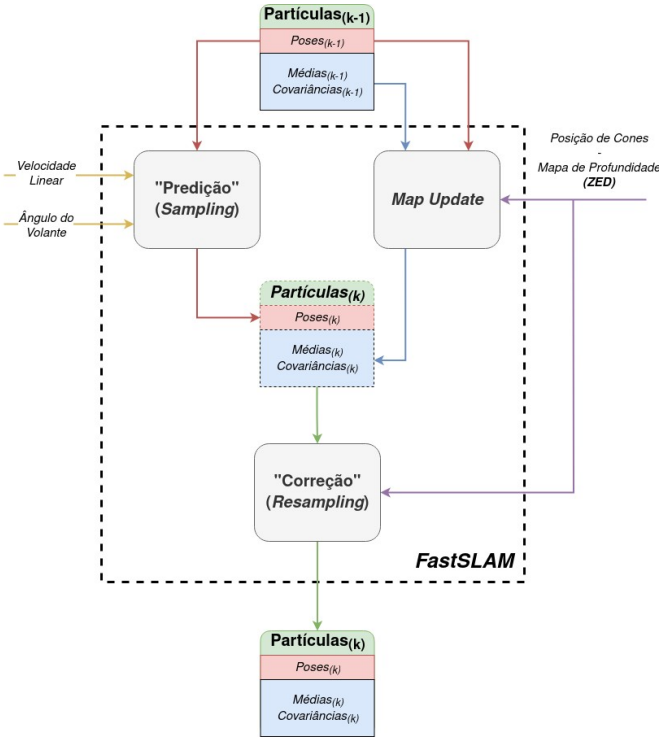

Figura 3 - Diagrama mostrando os diferentes componentes do FastSLAM e o fluxo de informação entre eles.

O programa criado foi divido em três componentes associados às etapas de *sampling*, *map update* e *resampling*. O fluxo de dados entre eles é representado graficamente na Figura 3. Ela mostra a sua independência: eles requerem apenas os parâmetros exibidos e um conjunto de partículas qualquer, portanto podem ser executados em qualquer ordem. Dessa forma, o componente de *sampling* pode ser executado

sempre que dados de velocidade linear e ângulo de volante são recebidos e o componente *map update* sempre que uma nova medição da localização de um cone é recebida. O componente de *resampling*, no entanto, pode ser executado em qualquer momento e a qualquer frequência que o projetista desejar [7].

Para testar os algoritmos, optou-se por usar um simulador desenvolvido na equipe de fórmula estudantil UNICAMP ERacing. Ele fornece dados ”ideais” de velocidade linear, ângulo do volante e localização de *landmarks*, porém somados a valores gerados aleatoriamente de forma a simular o ruído presente na leitura dos sensores que seriam utilizados em experimentos.

RESULTADOS E DISCUSSÃO:

Um dos mapas gerados a partir de dados ruidosos pelo algoritmo desenvolvido foi o da pista da prova de *skidpad*, realizado no campeonato *Formula Student Germany*. O resultado é apresentado na Figura 3, em que a localização real dos cones está marcada em laranja e a estimativa de sua localização está marcada em azul.

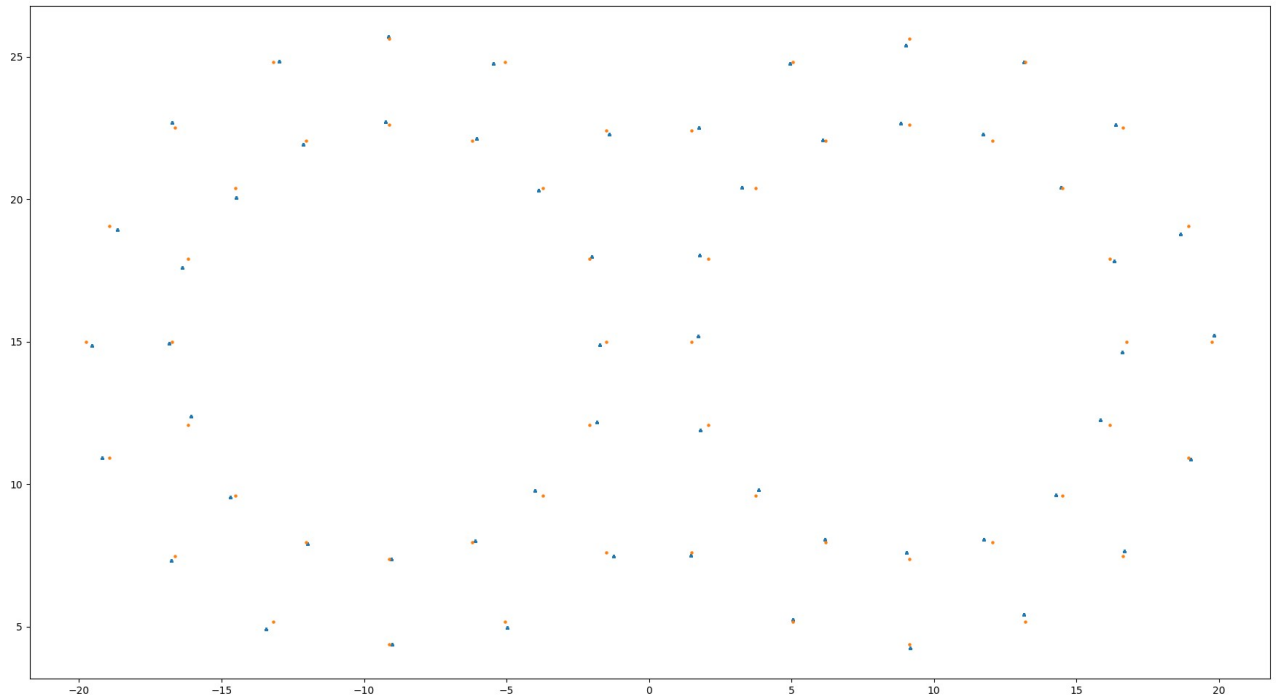

Figura 4: Mapa gerado pelo FastSLAM 1.0 (azul) sobreposto ao “ground truth” (laranja).

Observa-se que, de modo geral, há pouca diferença entre as localizações dos cones estimadas pelo FastSLAM e suas localizações reais. Além disso, o erro RMS dessas estimativas é de aproximadamente 0,16m. Dessa forma, é possível que o algoritmo desenvolvido seja adequado para uso na *Formula Student Germany* ou em outra competição similar na categoria de veículos autônomos. No entanto, para verificá-lo é necessário realizar testes a partir de dados experimentais.

CONCLUSÃO:

Diversos algoritmos de SLAM foram usadas com sucesso em competições de formula student, entre elas variações do FastSLAM. No entanto, para verificar a efetividade do algoritmo desenvolvido são necessários testes com dados experimentais. Além disso, existem possibilidades de melhoria do algoritmo especialmente no quesito de eficiência computacional.

Como trabalhos futuros, pretende-se testar a possibilidade de deixar de executar o componente *map update* assim que o mapa gerado atingisse um nível de qualidade satisfatório. Dessa forma reduzir-se-á a exigência computacional do sistema autônomo, o que possivelmente aumentaria sua efetividade.

Além disso, uma melhoria a ser feita é a utilização de mais sensores a fim de aumentar a exatidão e precisão dos mapas gerados. Alguns dos sensores para esse fim que poderiam ser utilizados são LIDAR's, IMU's e GPS's.

BIBLIOGRAFIA

1 BETZ, J. et al. Autonomous Vehicles on the Edge: A Survey on Autonomous Vehicle Racing. **IEEE Open Journal of Intelligent Transportation Systems**, v. 3, p. 458–488, 2022.

2 AVIZZANO, C. A. et al. **Visual SLAM for Driverless Racing Vehicle**. Dissetação (Mestrado em Sistemas Computacionais Embarcados) – Università di Pisa, Pisa, 2019.

3 ZEILINGER, MARCEL et al. Design of an Autonomous Race Car for the Formula Student Driverless (FSD). OAGM&ARW Joint Workshop, 2017. **Anais…** Viena: 2017.

4 KABZAN, J. et al. AMZ Driverless: The full autonomous racing system. **Journal of Field Robotics**, [*S.l.*], v. 37, n. 7, p. 1267–1294, 2020.

5 ANDRESEN, L. et al. Accurate Mapping and Planning for Autonomous Racing. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020. **Anais…** [*S.l.*], 24 out. 2020. Disponível em: <http://arxiv.org/abs/2003.05266>. Acesso em: 3 ago. 2025.

6 LARGE, N. L. **A comparison of different approaches to solve the SLAM problem on a Formula Student Driverless race car**. Dissertação (Mestrado) - Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, Karlsruhe, 2020.

7 THRUN, S. et al. **FastSLAM: An Efficient Solution to the Simultaneous Localization And Mapping Problem with Unknown Data Association**. [*S. l.*], 18 abr. 2004. Disponível em: <https://robots.stanford.edu/papers/Thrun03g.pdf>. Acesso em: 3 ago. 2025.