MapUpdater: Atualizador dey Mapas Baseado em Life Long SLAM

Luiz Rogério Araujo de Araujo
Programa de Pósgradução em Informática (PPGI)
Universidade Federal do Espírito Santo (UFES)
Vitória, ES, Brasil
luiz.r.araujo@edu.ufes.br

Agnelo Pereira Lima Junior
Programa de Pósgradução em Informática (PPGI)
Universidade Federal do Espírito Santo (UFES)
Vitória, ES, Brasil
agnjuniorlima@gmail.com

Aurea Oliveira
D*epartamento de Informática (DI)*
Universidade Federal do Espírito Santo (UFES)
Vitoria, ES, Brasil
aurea.s.oliveira@edu.ufes.br

Miguel Gewehr de Oliveira
D*epartamento de Informática (DI)*
Universidade Federal do Espírito Santo (UFES)
Vitória, ES, Brasil
miguel.g.oliveira@edu.ufes.br

***Resumo*— Este trabalho visa a implementação de um novo modulo atualizador de mapas, denominado MAPUPDATER, que visa à implementação de um sistema que permite atualização precisa do *offline map* (mapa inicial), em especial a retirada ou inclusão de objetos (carros estacionados, barrancos, obstáculos, árvores derrubadas, colisões entre veículos, etc.) de forma a proporcionar situações mais próximas da realidade, ao mesmo tempo, garantir uma navegação segura e eficiente ao longo do tempo no sistema de controle de veículos autônomos do projeto denominado IARA (Intelligent Autonomous Robotic Automobile), desenvolvido pelo Laboratório de Computação de Alto Desempenho (LCAD) da Universidade Federal do Espírito Santo (UFES), cujo código de controle é baseado no CARMEN (Carnegie Mellon Robot Navigation Toolkit), cuja versão foi denominada CARMEN-LCAD. É apresentado um procedimento para edição (alterações) do *offline map*, que é executado antes da execução do CARMEN (independente), outro procedimento para alteração (edição) do mapa atual durante a execução do CARMEN e um programa para geração de mapa a partir do *log*.**

Keywords—Laser scanning, place recognition, bag of words, rasterization, mapping, simultaneous localization and mapping.

I. INTRODUÇÃO

O pacote CARMEN (Carnegie Mellon Robot Navigation Toolkit ou Car Autonomous Robot for Mobile Environment Navigation) é uma ferramenta modular e extensível de código aberto e livre para controle e navegação de robôs e veículos móveis. Ele foi desenvolvido por instituições de pesquisa para facilitar a implementação de sistemas de navegação autônomos em veículos e robôs. Os principais recursos e funcionalidades do CARMEN são:

1) **Controle, Atuação e Sensores:** O CARMEN possui módulos para controle e atuação para movimentação do robô, sensores como: LiDAR (*Light Detection anda Ranging*), câmeras, odômetros, dentre outros, que permitem precisão na sua localização, movimentos e percepção do ambiente ao seu redor (mapeamento).
2) **Localização e Mapeamento:** O pacote fornece ferramentas para localização e mapeamento (SLAM, *Lifelong SLAM versão 1.0 e 2.0*), o que permite que o robô crie e atualize continuamente mapas do ambiente e se localize dentro deles.
3) **Evitação de Obstáculos:** Inclui algoritmos para evitar obstáculos, garantindo que o robô possa navegar em ambientes complexos sem colidir com objetos.
4) **Registro de mapas e localização (estados e poses):** Inclui funcionalidades para registro e armazenamento de poses e mapas em arquivos do tipo *log*. Inclusive, tem a capacidade de fazer simulações (playback) a apartir dos arquivos de log.
5) **Rastreamento de Pessoas:** Inclui funcionalidades para rastreamento de pessoas, o que pode ser útil em aplicações de robótica social e segurança.
6) **Planejamento de Caminhos:** Possui módulos para planejamento de caminhos, ajudando o robô a encontrar a melhor rota para alcançar um destino específico.
7) **Modularidade e Extensibilidade:** É extremamente modular e extensível, permitindo que desenvolvedores adicionem novos módulos, funcionalidades e/ou melhorias conforme suas necessidades ou conveniência.

O pacote CARMEN é utilizado em diversos projetos de pesquisa, desenvolvimento e testes de veículos autônomos, inclusive, foi usado no projeto IARA (*Intelligent Autonomous Robotic Automobile*), desenvolvido pelo Laboratório de Computação de Alto Desempenho (LCAD), ligado ao Prograna de Pós-graduação em Informática (PPGI), da Universidade Federal do Espírito Santo (UFES), cujo código aberto está no endereço https://github.com/LCAD-UFES/carmen_lcad.

A localização e mapeamento simultâneo - Lifelong SLAM (Simultaneous Localization and Mapping) - é uma abordagem do CARMEN para fazer o mapeamento do ambiente e ao

mesmo tempo estimar a localização dos robôs e veículos autônomos, continuamente, em ambientes dinâmicos e de longo prazo. Essa abordagem enfrenta desafios únicos, como a necessidade de manter a precisão e eficiência ao longo do tempo, mesmo quando o ambiente muda, o que pode afertar também a localização do robô [01].

O módulo MAPUPDATER implementa o Lifelong SLAM (*Simultaneous Localization and Mapping*), que é uma abordagem avançada de SLAM projetada para permitir que robôs e sistemas autônomos operem de forma contínua e eficiente em ambientes dinâmicos e de longo prazo [01], [03]. O SLAM é essencial para o funcionamento de carros e robôs autônomos.

No caso do MAPUPDATER, o SLAM é feito com a operação do localizador Ackerman (l*ocalizer_ackerman*) e do servidor de mapas de localização (*map_server*), simultaneamente com a operação do mapeamento através do MAPUPDATER. Ackerman é um conceito de geometria de direção que ajuda os veículos a girar de forma eficiente e suave; baseia-se na ideia de que as rodas internas e externas de um veículo devem ter ângulos de direção diferentes ao girar. Pode significar um quadrilátero que combina partes da suspensão do veículo, formando ângulos e distâncias que se alteram de acordo com o movimento do volante.

Lifelong SLAM e SLAM tradicional têm objetivos semelhantes (Cap. 13, [05]), mas diferem significativamente em suas abordagens e capacidades, as principais diferenças são:

1. **Continuidade e Adaptação**:
   - **SLAM tradicional**: Foca em mapear e localizar em um ambiente estático ou que muda pouco durante um curto período de tempo. Uma vez que o mapa é criado, ele não é atualizado com novas informações.
   - **Lifelong SLAM**: Projetado para operar continuamente em ambientes dinâmicos e de longo prazo. Ele atualiza o mapa com novas informações ao longo do tempo, permitindo que o robô se adapte a mudanças no ambiente
2. **Gerenciamento de Dados**:
   - **SLAM tradicional**: Pode sofrer com o crescimento contínuo do gráfico de poses, levando a uma perda de eficiência e aumento da complexidade computacional.
   - **Lifelong SLAM**: Utiliza técnicas de poda de gráficos e esparsificação para manter o tamanho do gráfico gerenciável, preservando a precisão e eficiência
3. **Robustez a Mudanças**:
   - **SLAM tradicional**: Pode ter dificuldades em ambientes que mudam frequentemente, como áreas públicas ou industriais.
   - **Lifelong SLAM**: É mais robusto a mudanças frequentes e pode lidar com revisitas a locais já mapeados, ajustando o mapa conforme necessário
4. **Aplicações**:
   - **SLAM tradicional**: Ideal para aplicações onde o ambiente é relativamente estável, como robôs de serviço em ambientes controlados.
   - **Lifelong SLAM**: Adequado para aplicações em ambientes dinâmicos, como centros comerciais, armazéns e áreas urbanas, onde o ambiente muda constantemente

Essas diferenças tornam o Lifelong SLAM uma escolha superior para robôs que precisam operar de forma eficiente em ambientes que mudam ao longo do tempo. Está disponível na versão 1.0 e 2.0, sendo essa última recomendada para casos em que a imprecisão do controle é maior que a precisão das medidas conforme Cap 13 de [05].

A estrutura geral do **Lifelong Localization and Mapping** (SLAM) em ambientes dinâmicos é projetada para permitir que robôs mantenham um mapa atualizado e preciso ao longo do tempo, mesmo quando o ambiente muda. Os principais componentes de sua estrutura são [04] :

1. **Representação de Múltiplas Sessões**: O mapa global é dividido em várias sessões, cada uma representando um período específico ou uma mudança significativa no ambiente. Isso permite que o sistema mantenha um histórico das mudanças e atualize o mapa conforme necessário
2. **Estratégia de Atualização de Mapas**: Inclui a construção de mapas, refinamento do gráfico de poses e esparsificação. O sistema atualiza continuamente o mapa com novas informações, garantindo que ele reflita com precisão o ambiente atual.
3. **Refinamento e Esparsificação do Gráfico de Poses**: Para evitar o aumento ilimitado do uso de memória, o sistema utiliza um método de poda de mapas baseado na árvore de Chow-Liu, que maximiza a informação mútua. Isso ajuda a manter a complexidade computacional sob controle enquanto preserva a precisão do mapa
4. **Validação e Testes**: Essa estrutura foi validada através de testes extensivos em ambientes reais, como supermercados, onde o robô foi implantado por mais de um mês. Esses testes demonstraram a eficácia do sistema em manter um mapa atualizado e preciso, mesmo em ambientes complexos e dinâmicos.

O CARMEN requer um mapa inicial do ambiente que é gerada pelos sensores do robô ou veículo autônomo, assim que começa a operar (acorda), que é chamado de *offline map*.

Os mapas são gerados através dos sensores dos robôs ou veículos autônomos e a técnica de criação de mapas deriva dos mapas de remissão (*remission map*), que se refere aos mapas que registram a intensidade de reflexões de sensores, como LiDAR ou radares. Esses mapas são usados para melhorar a precisão da localização e mapeamento, especialmente em ambientes complexos, são particularmente úteis em ambientes industriais e urbanos, onde há uma variedade de materiais e superfícies que podem afetar a precisão dos sensores.

A geração dos mapas em geral é feita através do módulo *mapper.h* do CARMEN, que trata as informações do mapa com a seguinte estrutura de dados definida no *grid_mapping.h*:

```
typedef struct { carmen_map_t *offline_map;
                 carmen_map_t*occupancy_map;
                 carmen_map_t *sum_occupancy_map;
```

```
        carmen_map_t *count_occupancy_map;
        carmen_map_t *new_occupancy_map;
        carmen_map_t *new_sum_occupancy_map;
        carmen_map_t *new_count_occupancy_map;
        carmen_map_t *sum_remission_map;
        carmen_map_t *sum_sqr_remission_map;
        carmen_map_t *count_remission_map;
        carmen_map_t *new_sum_remission_map;
        carmen_map_t *new_sum_sqr_remission_map;
        carmen_map_t *new_count_remission_map;
        carmen_map_t *snapshot_map;
        carmen_map_t *sum_remission_snapshot_map;
        carmen_map_t
        *sum_sqr_remission_snapshot_map;
        carmen_map_t
        *count_remission_snapshot_map;
        carmen_map_t *log_odds_snapshot_map;
        carmen_map_t *moving_objects_raw_map;
        carmen_map_t
        *map_buffer[MAP_BUFFER_SIZE + 1];
        int map_buffer_index;
        double x_origin;
        double y_origin;
} carmen_map_set_t;
```

Cada célula do *offline map* pode ter uma dada probabilidade (valor de 0.0 a 1.0), ou uma probabilidade desconhecida (-1.0), ou ainda uma marcação de ocupação manual (-2.0). Além disso, a cada célula é associada: (i) a uma contagem de quantas vezes ela foi atualizada, (ii) a soma dos valores de intensidade de reflexão do laser (*remission map*) que ela apresentou, e (iii) a soma dos quadrados do *remission* que ela apresentou - estes três valores são usados para computar o mapa calibrado de *remission*.

Se o parâmetro de entrada *mapper_mapping_mode_on_create_map_sum_and_count estiver "on", também é associada a cada célula do offline map a contagem de quantas vezes cada célula foi acessada, assim como a soma da probabilidade de ocupação de cada célula acessada.*

No MAPUPDATER, a soma da probabilidade de ocupação de cada célula acessada é igual à soma dos log_odds, de forma que esse último no offline map não poderá ultrapassar um valor máximo (MAX_LOG_ODDS), ou ser inferior a um valor mínimo (MAX_LOG_ODDS). Além disso, a forma de contagem de quantas vezes cada célula foi acessada é modificada por um incremento de um fator de atualização da célula que é proporcional à velocidade do veículo, denominado strenght_factor - quando a velocidade for inferior à um limiar (0.5 m/s, por exemplo), este fator será igual a zero. Quando a velocidade aumentar, este fator aumenta também para tornar as medidas das taxas de atualização das células proporcionais entre si, independente da velocidade. Assim, cada célula passa a ter uma medida do strength de seu log_odds atual equivalente à soma dos strenght_factor acumulados nela.

A) Atualização das células do offline map pelo MAPUPDATER

Inicialmente, o offline map precisa ser cuidadosamente criado com o graphslam ou pelo fastslam, como já é feito atualmente, mas sem que nenhuma limpeza manual do mesmo precise ser feita.

Depois do seu processo de criação, suas células podem ser atualizadas pelo MAPUPDATER das seguintes formas:

1. *Uma célula apresenta uma alta probabilidade de ocupação (próxima de 1.0), mas esta probabilidade deve ser reduzida para um valor menor que 0.5.*
2. *Uma célula apresenta uma baixa probabilidade de ocupação (próxima de 0.0), mas esta probabilidade deve ser aumentada para um valor maior que 0.5.*
3. *Uma célula deve manter a sua probabilidade de ocupação aproximadamente como está.*
4. *Uma célula possuía um valor de probabilidade desconhecido (-1.0), mas deve receber um valor de probabilidade entre 0.0 e 1.0.*

Para atualizar o offline map, o MAPUPDATER computa um online map ao mesmo tempo em que o veículo opera no ambiente. Este online map deve incluir a soma dos log_odds e o strength dos log_odds de cada célula do mapa.

B) Forma 1 - Remoção de pontos ocupados no offline map.

Se o log_odds de uma célula de um bloco do online map corrente, indica probabilidade maior que 0.5 no offline map, leva à uma probabilidade menor que 0.5 e o strength da mesma no online map for maior que o strength da célula do offline map, a probabilidade da célula do offline map é atualizada segundo o log_odds da célula do online map, reduzindo sua probabilidade para abaixo de 0.5.

O strength inicial das células do offline map é aquele observado quando da criação do offline map, ou é igual à uma constante global denominada MAX_CELL_STRENGTH, o que for menor. Se a célula do offline map tiver seu *log_odds* atualizado, seu s*trength* é atualizado também de acordo com o valor computado para a célula do *online map* (se o strength for maior que MAX_CELL_STRENGTH, a célula recebe MAX_CELL_STRENGTH). Note que o strength das células do *online map* pode ultrapassar MAX_CELL_STRENGTH.

C) Forma 2 - Adição de pontos ocupados no *offline map*

A Forma 2 é muito similar à Forma 1. Nela, se o *log_odds* de uma célula do *online map*, correntemente com probabilidade menor que 0.5 no *offline map*, leva à uma probabilidade maior que 0.5 e o strength da mesma no *online map* for maior que o *strength* da célula no *offline map*, a probabilidade da célula do *offline map* é atualizada segundo o *log_odds* da célula do *online map*, aumentando a sua probabilidade para acima de 0.5.

O strength das células do *offline map* é inicializado e atualizado do mesmo modo que na Forma 1.

D) Forma 3 - Reforço do estado de longo prazo do *offline map*

Se o *log_odds* de uma célula do *online map* leva à uma probabilidade próxima (no mesmo lado, a contar de 0.5) à do *offline map*, o *log_odds* da célula do *online map* é somado ao *log_odds* da célula no *offline map* e o *strength* da célula no *online map* é somado ao strength da célula no *offline map*. Se o *strength* ultrapassar MAX_CELL_STRENGTH, o valor armazenado na célula do *offline map* será MAX_CELL_STRENGTH. O *log_odds* da célula no *offline map* também será limitado à uma faixa que vai de -MAX_LOG_ODDS a MAX_LOG_ODDS.

Esta Forma 3 leva à consolidação do *offline map* com o tempo.

E) Forma 4 - Adição de novos pontos ocupados e livres no *offline map*

Se uma célula do *offline map* nunca foi acessada previamente e ela é acessada no *online map*, a probabilidade da célula no *offline map* é atualizada segundo o *log_odds* do *online map* e o *strength* da célula no *offline map* é atualizado de acordo com o correntemente computado para a célula no *online map* (se o *strength* for maior que MAX_CELL_STRENGTH, a célula do *offline map* recebe MAX_CELL_STRENGTH).

Veículos ou pessoas se movendo na área de operação do veículo autônomo baseado no CARMEN-LCAD e com MAPUPDATER não interferiram no *offline map*, já que a medida do *strength* das células por onde passaram no *online map* não seria alta o suficiente para ultrapassar o *strength* das células correspondentes do *offline map*.

Pode haver casos, contudo, onde objetos móveis venham /a escapar dos limiares impostos por MAX_CELL_STRENGTH, -MAX_LOG_ODDS e MAX_LOG_ODDS. Nestes casos ainda será necessária a intervenção humana. Espera-se, entretanto, que seja possível um bom ajuste destes parâmetros de modo a permitir longos períodos de operação sem intervenção humana.

F) Execução dos Procedimentos do MAPUPDATER

Alteração do *offline map* pode ser feita antes da execução do CARMEN e durante a execução do CARMEN, através de dois procedimentos distintos. A alteração MAISé feita através da edição do *offline map*, através de módulo de edição (*map_editor*) que deve ser executado antes da execução do CARMEN.

O procedimento para atualização do mapa atual durante a execução do CARMEN, é mais complexo, é feito através do módulo *mapper*. Para isso, adotamos a infraestrutura de mapeamento voltada para o salvamento de mapas quando o mapper está sendo usado no modo *mapping_mode on*. Além disso, desativamos o *map decay*, entre outras facilidades.

O procedimento de edição do mapa atual, durante a execução do CARMEN ainda não foi concluído, uma vez que se faz necessário alterar a interface do CARMEN com o usuário, para incluir um botão acionador para interromper a execução do CARMEN, apresentar o editor de mapa, ao final da edição, salvar o novo mapa atual e, então continuar a execução a partir do novo mapa editado. Alguns cuidados dever ser tomados para não alterar o local do carro ou robô autônomo, dentre outras restrições.

Ou seja, o CARMEN-LCAD executará o *mapper*, que por sua vez salva os mapas no mesmo local de onde o *map_server* busca-os. Deste modo, depois que o MAPUPDATER atualiza um bloco de um mapa, o *map_server* usa a versão nova do mapa (atualizada) ao invés da mais velha, inclusive do offline map.

A implementação atual é o FASTSLAM 2.0 para mapas de grades (*grids*) que é um algoritmo de SLAM *online* para geração de mapas, que considera que o último estado do robôs, contém todas as informações sobre a pose do robô e o mapa (modelo de Markov), ao contrário do GraphSLAM, que é um algoritmo de full SLAM *offline*, que considera todos os estados do *log,* apenas seu *time stamp e* não simultaneamente com os dados do Velodyne, para fazer o mapa e saber onde o robô estava no mapa (localização).

O FastSLAM cria apenas o mapa de ocupação (*occupancy*) e não cria o mapa de *remission* e outros mapas associados.

Assim, o estado inicial do robô é muito importante para a construção de um bom mapa de ocupação, sendo conveniente que o robô equipado com bom GPS esteja parado por pelo menos 6 segundos, de forma à assegurar a melhor precisão do mapa e maior precisão da localização do robô. Importante também considerar a orientação do GPS, sendo que se for usado GPS sem orientação, deve ser indicado o ângulo inicial do robô na variável *initial_angle* do FASTSLAM.

A odometria deve estar bem calibrada, assim como a posição dos sensores (no arquivo de inicialização) e, especialmente, a posição da *sensor_board*, Velodyne e GPS. Melhor colocar o ângulo de *pitch* e *yaw* no Velodyne, e não na *sensor_board*.

II. MÉTODO PROPOSTO

A. Levantamento Bibliográfico sobre mapeamento e localização (Lifelong SLAM) - Inicialmente foram feitos estudos da teoria relacionada ao mapeamento e localização com base na obra [05].

B. Levantamento da documentação e do código do CARMEN. Por segundo, foi analisada a documentação referente CARMEN-LCAD, inclusive, o estado da arte do código fonte na UFES.

C. Método Experimental (a seguir).

D. Elaboração da documentação e relatório no Github.

III. MÉTODO EXPERIMENTAL

A. Instalação do CARMEN-LCAD nos computadores equipados com GPU - Foi instalado o CARMEN-LCAD num computador de mesa e em dois notebooks equipados com GPU NVIDIA. A instalação requer tempo para baixar os códigos e as bibliotecas necessárias para execução do código, como é caso do Python, C, CUDA (executado na GPU), dentre outras. A instalação do CARMEN foi feita no sistema operacional Ubuntu 20.04 em todos os computadores, conforme instruções no endereço https://github.com/LCAD-UFES/carmen_lcad.

B. Obtenção dos 33 Arquivos (logs) das voltas da IARA em torno da UFES gerados pelo CARMEN-LCAD, durante as voltas da IARA em torno da UFES.

C. Escolha do arquivo de log para testes de execução do CARMEN-LCAD e novo móulo FASTSLAM- Foi escolhido o menor arquivo de log para testes e execução do código no modo playback do CARMEN-LCAD.

D. Incorporação dos módulos atualizados do FASTSLAM ao CARMEN-LCAD.

E. Preparação dos arquivos de inicilização do CARMEN

F. Execução do CARMEN-LCAD no modo playback usando arquivo de log de uma volta da UFES.

G. Preparação da documentação Github

IV. RESULTADOS EXPERIMENTAIS

Os dois procedimentos para alteração do mapa citados no item F da seção anterior estão apresentados, documentados em https://github.com/agnjuniorlima/map-updatea. Foram executados logs da IARA em torno da UFES, que foram juntados (merge) para verificar a execução do MAPUPDATER. De outra forma, foram usados procedimentos de edição de mapa (alterações de regiões) antes e durante a execução do CARMEN.

Entre as dificuldades, destacamos: (1) problemas técnicos relacionados à compilação e execução do CARMEN; (2) volume, complexidade documentação heterogênea (muito módulos sem comentários) dos módulos do CARMEN (afetou sensivelmente curva de aprendizado); (3) Problemas pessoais dos integrantes que reduziram a disponibilidade.

Sugerimos melhoria de suporte do suporte, aumentar o nível de da documentação, uso de ferramentas de gerenciamento do ciclo de vida e documentação.

No que se refere às atividades, sugerimos a conclusão da função de edição de de mapas antes e depois da execução do CARMEN através do menu de navegação.

AGRADECIMENTOS

Nossos mais sinceros agradecimentos aos nossos companheiros (esposas ou maridos), filhos, pais, irmãos, tios, familiares, amigos e colegas da UFES e de trabalho pelo estímulo e apoio incondicional nos momentos mais difíceis ao longo dessa jornada. Agradecimento muito especial para a Profa. Claudine Santos Badue pela dedicação, disponibilidade, pela pronta atenção e auxílio nas atividades, pela exposição incorrigível do conteúdo da disciplina Robótica Probalística, assunto que domina com maestria e segurança absoluta, pelo provimento dos recursos tecnológicos e administração do LCAD. Indispensável também agradecer ao Professor Alberto Ferreira Souza pela sua dedicação, conhecimento, disponibilidade, liderança e apoio incondicional nas atividades relacionadas a IA e Robótica, inclusive, pesquisa e desenvolvimento relacionadas à GPT.

REFERÊNCIAS

[1] W. Ali, P. Liu, R. Ying and Z. Gong, “A life-log SLAM approach using adaptable local maps baseado n rasterized LIDAR images,” IEEE SENSOR JOURNAL, Vol XX, NO. XX, Jul 2021 - .

[2] S. Thun, W. Burgard, D. F//ox, “Probabilistic Robotics”, The MIT Press, Cambridge, Massachusetts, London, 2006.

[3] W. Ali, P. Liu, R. Ying and Z. Gong, “A life-long SLAM approach using adaptable local maps based on rasterized LIDAR images”

[4] M. Zhao, X. Guo, L. Song, B. Qin, X. Shi G. Lee G. Sun, “A General Framework for Lifelong Localization and Mapping in Changing Enviroment”