Mapeamento e Localização Simultâneos em Ambientes Dinâmicos usando Detecção de Pessoas

João Carlos Virgolino Soares[1], **Marcelo Gattass**[2] **(Co-orientador),**
Marco Antonio Meggiolaro[1] **(Orientador)**

[1]Departamento de Engenharia Mecânica – Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)

[2]Departamento de Informática – Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)

virgolinosoares@gmail.com, mgattass@tecgraf.puc-rio.br, meggi@puc-rio.br

Abstract. *Simultaneous Localization and Mapping is a fundamental problem in mobile robotics. However, the majority of Visual SLAM algorithms assume a static scenario, limiting their applicability in real-world environments. Dealing with dynamic content in Visual SLAM is still an open problem. This work presents the first visual SLAM method for crowded human environments using people detection.*

Resumo. *Localização e Mapeamento Simultâneos é um problema fundamental em robótica móvel. No entanto, a maioria dos algoritmos de SLAM Visual assume um cenário estático, limitando sua aplicabilidade em ambientes do mundo real. Lidar com conteúdo dinâmico em SLAM visual ainda é um problema em aberto. Este trabalho apresenta o primeiro método de SLAM visual feito para ambientes humanos lotados usando detecção de pessoas.*

Submissão ao CTDR (PhD).

Data de conclusão do doutorado: 11 de Maio de 2022.

Este trabalho é um resumo de uma das contribuições da tese de João Soares, que pode ser acessada em:
https://www.maxwell.vrac.puc-rio.br/59878/59878.PDF

1. Introdução

O problema de localização e mapeamento simultâneos (SLAM) é fundamental em robótica móvel, especialmente em ambientes desconhecidos e não estruturados, sendo pré-requisito para diversas tarefas, como a navegação. Esse problema consiste em criar um mapa do ambiente e, simultaneamente, estimar a pose do robô no mapa criado.

Câmeras são uma escolha comum como sensor em um sistema SLAM devido ao seu baixo custo e riqueza de informações, permitindo sistemas de odometria visual precisos, algoritmos de *loop closure* robustos e uso de informações semânticas, entre outros benefícios. As câmeras RGB-D têm a vantagem extra de fornecer também informações de profundidade. O problema de SLAM visual consiste em realizar localização e mapeamento usando apenas uma câmera.

Existem vários sistemas de SLAM visual na literatura, com alta precisão e eficiência. Entre eles podemos citar ORB-SLAM2 [Mur-Artal and Tardós 2017], LSD-SLAM [Engel et al. 2014], RGBDSLAM [Endres et al. 2014] e RTAB-Map [Labbé and Michaud 2019]. No entanto, a maioria desses sistemas assume um ambiente estático, o que impõe uma limitação de sua aplicabilidade em cenários do mundo real.

Os principais desafios na execução de SLAM em ambientes dinâmicos são: detectar objetos dinâmicos na cena, evitar que esses objetos sejam rastreados e excluí-los do mapa. Alguns sistemas SLAM que funcionam em ambientes dinâmicos utilizam abordagens puramente geométricas para detectar objetos em movimento. No entanto, eles geralmente não conseguem detectar a presença de objetos dinâmicos *a priori*. Pessoas, por exemplo, quando estão inicialmente paradas, podem gerar desvios de odometria ou erros de *loop closure* a longo prazo. Tarefas de visão computacional, como detecção de objetos e segmentação de instâncias, fornecem informações semânticas da cena que permitem o reconhecimento de tais objetos.

Há um número crescente de sistemas de SLAM visual contando com detectores de objetos de aprendizado profundo para filtrar o conteúdo dinâmico das imagens. No entanto, eles também não são eficientes nem adequados para trabalhar com uma quantidade grande de objetos dinâmicos, como uma multidão de pessoas, por exemplo. O principal objetivo deste trabalho é criar um sistema SLAM capaz de operar em ambientes lotados em tempo real, usando um rede personalizada YOLO Tiny especializada em detecção de pessoas em multidões e um algoritmo para filtragem de *keypoints*.

Assim, este trabalho propõe Crowd-SLAM, o primeiro sistema de SLAM visual para ambientes lotados de pessoas. A detecção de objetos YOLO é usada para filtrar as pessoas na cena. A rede YOLO é treinada usando um conjunto de dados para ambientes lotados, alcançando um desempenho em tempo real com alta precisão. A metodologia proposta é avaliada usando vários conjuntos de dados e comparada com sistemas estado-da-arte.

2. Revisão Bibliográfica

2.1. SLAM visual em Ambientes Dinâmicos

A maioria dos sistemas estado-da-arte de SLAM visual foram projetados com uma premissa de ambiente estático. Portanto, eles não são capazes de lidar com cenários dinâmicos. Os que lidam com conteúdo dinâmico na cena geralmente o tratam como ruído e o filtram usando métodos diretos ou baseados em *features*.

DynaSLAM [Bescos et al. 2018] usa o método de segmentação de instâncias chamado Mask R-CNN [He et al. 2017] para obter a informação dos pixels das pessoas na cena, usando-a para filtrar características dinâmicas *a priori*. Apesar de sua alta precisão e robustez, DynaSLAM não pode funcionar em tempo real devido à alta exigência computacional do Mask R-CNN. Por outro lado, a detecção de objetos é uma tarefa que pode ser realizada em tempo real, dependendo da técnica utilizada.

2.2. Detecção de Objetos

Detecção de objetos é a tarefa de determinar a localização, caixa delimitadora e classe de objetos em uma imagem. Existem diferentes tipos de algoritmos de detecção de objetos baseados em aprendizado profundo. Os detectores R-CNN, por exemplo, como Fast

R-CNN [Girshick 2015] e Faster R-CNN [Ren et al. 2015], usam um algoritmo para encontrar regiões potenciais para conter objetos e então usam uma rede neural convolucional (CNN) nessas regiões. Portanto, eles são conhecidos como detectores de dois estágios.

Apesar de serem precisos, esses algoritmos não funcionam em tempo real devido à complexidade de seus estágios. Por outro lado, detectores de estágio único, como o YOLO (*You Only Look Once*) [Redmon et al. 2016] e o SSD (*Single Shot Multibox Detector*) [Liu et al. 2016], são muito mais eficientes, pois utilizam apenas uma rede convolucional para obter tanto a caixa delimitadora e a classe do objeto sem a necessidade de gerar regiões candidatas, ao contrário dos detectores R-CNN. O YOLOv3 [Redmon and Farhadi 2018] pode rodar a 20 *frames* por segundo (FPS) em um computador com GPU, com 57,9 mAP, treinado com o dataset MS COCO [Lin et al. 2014].

2.3. SLAM visual em Ambientes Dinâmicos com Detecção de Objetos

A detecção de objetos baseada em aprendizado profundo tem sido amplamente aplicada em sistemas SLAM para filtrar objetos dinâmicos. Em Detect-SLAM [Zhong et al. 2018], a detecção de objetos SSD foi usada apenas em *keyframes* para superar o tempo de inferência lento de 0,31 s.

[Liu et al. 2019] usaram YOLOv3 combinado com *optical flow* para remoção de *keypoints* dinâmicas. No entanto, esse método não remove objetos dinâmicos *a priori*, potencialmente causando *loop closures* errados e desvios de odometria em uma cena com pessoas inicialmente estáticas. [Xiao et al. 2019] propuseram o Dynamic SLAM, que também usa detecção de objetos SSD para filtrar *keypoints* dinâmicos. Foi criado um módulo de correção semântica para criar uma máscara com o mesmo tamanho da imagem, para mapear pontos estáticos e dinâmicos, e um algoritmo de rastreamento seletivo para eliminar os objetos dinâmicos. No entanto, a criação da máscara pode ser computacionalmente exigente em imagens com alta resolução. Além disso, eles precisam da etapa adicional para filtragem de objetos dinâmicos que depende muito do número de objetos na cena. Nenhum dos trabalhos citados considera o cenário de ambiente lotado, o que pode levar a uma queda no desempenho ou precisão.

3. Metodologia

A Figura 1 mostra um diagrama da metodologia proposta, composta por quatro *threads*: Detecção de Objetos, Rastreamento, Mapeamento Local e *Loop Closure*. As quatro *threads* são executadas em paralelo. As imagens RGB são processadas nas *threads* de detecção e rastreamento de objetos simultaneamente. A *thread* de rastreamento extrai ORB *features* [Rublee et al. 2011] e aguarda as caixas delimitadoras fornecidas pela *thread* de detecção de objetos. Os *keypoints* e caixas delimitadoras são enviados para o filtro de *keypoints* dinâmicos e para o sistema de atualização do número de *keypoints* dentro da *thread* de rastreamento. O filtro dinâmico remove os pontos dinâmicos dentro das caixas delimitadoras, e o sistema de atualização altera o número de pontos proporcionalmente à área total filtrada, para evitar que o sistema perca o rastreamento.

3.1. SLAM

Este trabalho usa ORB-SLAM2 como solução global de SLAM. O ORB-SLAM2 tem três *threads* principais: rastreamento, *loop closure* e mapeamento local. O sistema proposto usa os mesmos sistemas de *loop closure* e mapeamento local do ORB-SLAM2. No

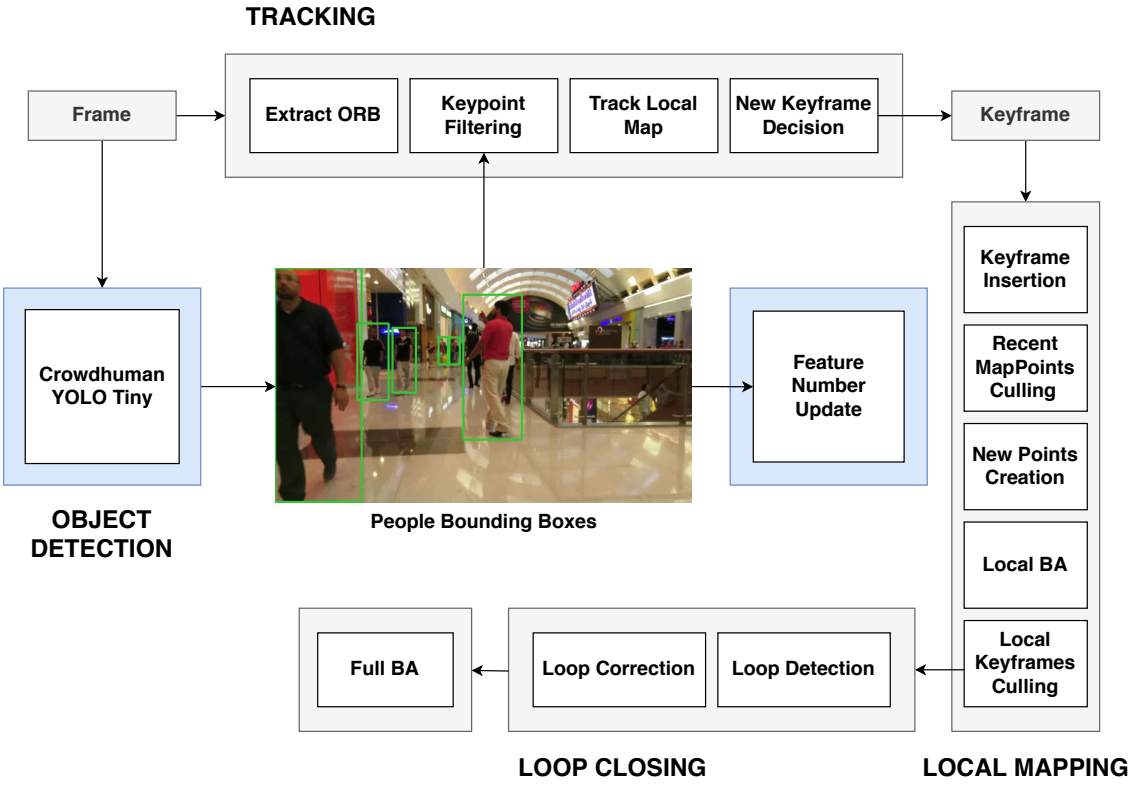


Figura 1. Diagrama da metodologia proposta

entanto, a *thread* de rastreamento foi modificada para incluir o algoritmo de remoção de pontos dinâmicos e o sistema de atualização do número de pontos. Além disso, uma nova *thread* foi adicionada para a detecção de objetos. ORB-SLAM2 funciona usando uma metodologia baseada em *keyframes*. A *thread* de rastreamento decide se cada nova imagem é uma *keyframe*. O sistema de *loop closure* compara as informações de cada nova *keyframe* com as anteriores, procurando por novos *loops* usando um módulo de reconhecimento baseado em DBoW2 [Galvez-López and Tardos 2012]. Uma vez que um *loop* é detectado, a trajetória da câmera é otimizada usando g2o [Kümmerle et al. 2011]. ORB-SLAM2 gera um mapa de nuvem de pontos esparsos e a trajetória otimizada da câmera.

3.2. Detecção de Pessoas

YOLOv3 fornece as classes dos objetos detectados, caixas delimitadoras 2D com suas posições correspondentes e um número de confiança para cada caixa. YOLOv3 Tiny é uma versão do YOLO com menos camadas e filtros, que possui velocidade de inferência 10 vezes maior. No entanto, tem uma precisão menor.

O MS COCO [Lin et al. 2014], usado no YOLOv3, possui 80 classes de objetos diferentes. No entanto, apenas a classe de pessoas é necessária neste trabalho. Propomos o *Crowdhuman YOLO Tiny* (CYTi), uma especialização do YOLO-Tiny que é mais precisa em ambientes lotados de pessoas. O CYTi é treinado com o *dataset Crowdhuman* [Shao et al. 2018], composto por mais de 20.000 fotos de pessoas em ambientes lotados, com 470.000 humanos e uma densidade média de 23 pessoas por imagem, muito mais do que outros conjuntos de dados de pessoas. Neste trabalho foram usadas 15.000 imagens

para treinamento e 4.370 para validação. O treinamento e validação foram realizados em uma GPU NVIDIA Quadro P2000 usando o framework Darknet [Redmon 2016].

O *dataset Multiple Object Tracking (MOT) Challenge* [Milan et al. 2016][Dendorfer et al. 2020] foi usado para avaliar as melhorias da rede treinada em ambientes lotados. Várias métricas são usadas para medir o desempenho de detecção. A precisão, mostrada na Eq. 1, é a taxa entre os verdadeiros positivos (TP) e o número total de detecções, que é a soma dos verdadeiros positivos e falsos positivos (FP).

$$Precision = \frac{TP}{(TP + FP)} \tag{1}$$

O *recall*, indicado na Eq. 2, é a taxa entre os verdadeiros positivos e a soma dos verdadeiros positivos e falsos negativos (FN).

$$Recall = \frac{TP}{(TP + FN)} \tag{2}$$

A Precisão de Detecção de Múltiplos Objetos (MODP) é dada por

$$MODP = \frac{\sum_{t=1}^{N_{frames}} \frac{OverlapRatio}{N_{mapped}(t)}}{N_{frames}} \tag{3}$$

onde N_{frames} é o número total de imagens, $N_{mapped}(t)$ é o número de objetos mapeados na imagem t, e o *OverlapRatio* é a soma da interseção sobre a união de cada objeto para cada imagem. A Precisão de Detecção de Múltiplos Objetos (MODA) é definida como

$$MODA = 1 - \frac{\sum_{i=1}^{N_{frames}} (m_i + fp_i)}{\sum_{i=1}^{N_{frames}} N_G^i} \tag{4}$$

onde m_i e fp_i são, respectivamente, as detecções perdidas e falsos positivos na imagem i, e N_G^i é o número de objetos na imagem i.

Duas sequências dos desafios MOT17 e MOT20 foram selecionadas: MOT20-01, MOT20-02, MOT17-09 e MOT17-11. As duas primeiras são cenas lotadas em uma estação de trem coberta, com uma câmera estática. O MOT17-11 é uma sequência em um shopping lotado com uma câmera em movimento. O MOT17-09 é uma cena ao ar livre lotada, com uma câmera estática perto das pessoas.

A Tabela 1 mostra os resultados da detecção usando o YOLO Tiny, YOLOv3 e CYTi para as sequências do desafio MOT. Além das métricas anteriores, ela também mostra o número total de verdadeiros positivos, falsos negativos e falsos positivos. Os símbolos de seta para baixo e para cima ao lado dos nomes das métricas significam que quanto menor ou maior o número, melhor é o desempenho geral da detecção.

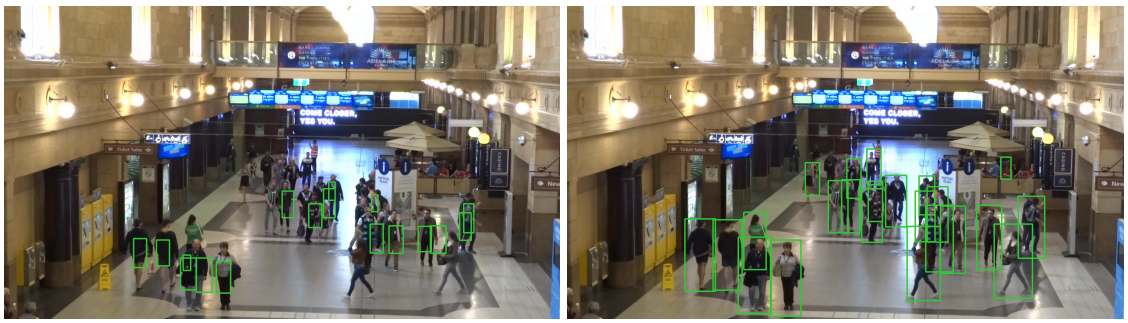
CYTi supera a rede YOLO Tiny em todas as métricas de todas as sequências. YOLO Tiny tem um desempenho ruim nessas sequências, principalmente no MOT20-01, não encontrando um único positivo verdadeiro. Os resultados da YOLOv3 também são

Tabela 1. Resultados de detecção de YOLO Tiny, YOLOv3 e CYTi

Sequências	Método	MODA ↑	MODP ↑	TP ↑	FN ↓	FP ↓	Prec. ↑	Recall ↑
	Tiny	-69.4	0.0	0	8924	6192	0.0	0.0
MOT20-01	YOLOv3	4.6	70.0	**6851**	**2144**	6437	51.6	**76.2**
	CYTi	**11.4**	**73.2**	6609	2441	**5575**	**54.2**	73.0
	Tiny	-67.4	64.2	78	20001	13612	0.4	0.6
MOT20-02	YOLOv3	8.1	69.7	**15971**	**4279**	14324	**78.9**	52.7
	CYTi	**14.0**	**73.4**	14872	5356	**12037**	73.5	**55.3**
	Tiny	-71.8	75.4	856	2184	3040	22.0	28.2
MOT17-09	YOLOv3	1.8	**77.4**	**2722**	**398**	2665	50.5	**87.2**
	CYTi	**60.6**	77.2	2528	559	**657**	**79.4**	81.9
	Tiny	-35.6	74.9	2064	3666	4104	33.5	36.0
MOT17-11	YOLOv3	29.5	**80.3**	**4917**	**1062**	3156	60.9	**82.2**
	CYTi	**52.4**	78.4	4786	1169	**1665**	**74.2**	80.4

comparados como referência. CYTi manteve a velocidade de inferência da YOLO Tiny, sendo mais de dez vezes mais rápido do que a YOLOv3, com uma maior precisão da detecção.

As Figuras 2a e 2b mostram, respectivamente, o resultado de detecção de objetos em uma cena lotada do MOT Challenge 2020 usando a rede YOLO Tiny e CYTi. A melhoria na precisão e exatidão é perceptível.



(a) YOLO Tiny **(b) CYTi**

Figura 2. Comparação de detecção de pessoas entre YOLO tiny e CYTi em um ambiente com muitas pessoas

3.3. Remoção de Outliers

Depois que as imagens passam pelo detector de pessoas, os *keypoints* que pertencem às pessoas são removidos da imagem. Ao contrário de outros sistemas, este algoritmo não necessita de uma máscara da imagem com informações sobre regiões estáticas e dinâmicas. Ele utiliza diretamente as caixas delimitadoras para realizar a filtragem.

As Figuras 3a e 3b mostram a detecção de *keypoints* do ORBSLAM2 e com o filtro de detecção de objetos proposto, respectivamente. Os filtros apagaram com sucesso

todos os pontos nas regiões com pessoas. Os pontos que aparecem na cadeira esquerda também são apagados, devido à fusão com a caixa delimitadora da pessoa, resultando em uma filtragem indireta de possíveis objetos dinâmicos.

(a) ORB-SLAM (b) Crowd-SLAM

Figura 3. Comparação de detecção de features

3.4. Atualização do Número de Features

Se um grande número de *keypoints* for filtrado de uma única imagem, as informações disponíveis para o sistema SLAM podem não ser suficientes para realizar o rastreamento. Dois problemas principais podem ocorrer simultaneamente ou independentemente. Primeiro, pode haver muitas pessoas na cena. Em segundo lugar, uma pessoa pode estar muito perto da câmera, ocupando a maior parte da imagem. Em ambas as situações, o problema não é o número de pessoas, mas a área total filtrada da imagem.

Mesmo com o sistema de relocalização robusto do ORBSLAM2, projetado especificamente para se recuperar de um rastreamento perdido, uma cena lotada pode impedir que o processo continue. Para superar esse problema, propomos um módulo para verificar a área filtrada e atualizar o número de ORB *features* detectados, em vez de definir um número alto estático. O número de *features* começa com um determinado valor inicial e aumenta em 300 para 30% da área filtrada, 500 para 60%, 700 para 90% e 1200 para mais de 95%. Este método beneficia o desempenho, pois mais *features* na imagem implica em mais esforço computacional, e simplesmente definir um valor estático alto deixaria o rastreamento mais lento sem necessidade no caso de não haver pessoas na cena.

As Figuras 4a a 4c mostram cenas com (a) nenhuma pessoa, (b) uma pessoa e (c) duas pessoas. O número de *keypoints* detectados é aumentado na terceira imagem devido ao aumento da área filtrada.

4. Resultados

4.1. Dataset TUM

O método proposto foi avaliado numericamente usando o *dataset* TUM RGB-D [Sturm et al. 2012]. Ele contém sequências de imagens RGB e de profundidade obtidas de uma câmera Microsoft Kinect, com *ground-truth*. Os dados foram gravados em 30Hz com resolução de 640 x 480.

(a) Sem pessoas na cena (b) Com uma pessoa na cena (c) Com duas pessoas na cena

Figura 4. Detecção do numero de features ORB

Dois tipos de sequências foram usados nesta avaliação. Nas sequências $fr3\_w$, duas pessoas estão andando na sala, movendo-se atrás de uma mesa, passando na frente da câmera e sentadas em cadeiras. Essas sequências são, portanto, altamente dinâmicas. As sequências $fr3\_s$ podem ser consideradas quase estáticas, pois as pessoas estão sentadas, fazendo movimentos principalmente com as mãos. Ambos os tipos são usados nesta avaliação.

Existem quatro tipos de movimento de câmera considerados: xyz, rpy, *half* e *static*. Para o movimento xyz, a câmera é movida ao longo dos três eixos, mantendo a mesma orientação. Na sequência rpy, a câmera é rotacionada em torno dos três eixos. Na sequência *half*, a câmera segue a trajetória de uma semi-esfera. Na sequência *static*, a câmera é mantida manualmente na mesma posição e orientação.

O Erro de Trajetória Absoluto (ATE) [Sturm et al. 2012] é utilizado para avaliar a consistência global da trajetória estimada, comparando as distâncias absolutas entre os componentes translacionais das trajetórias estimadas e o *ground-truth*. O cálculo do ATE é dado por

$$ATE_i = E_i^{-1} T G_i \tag{5}$$

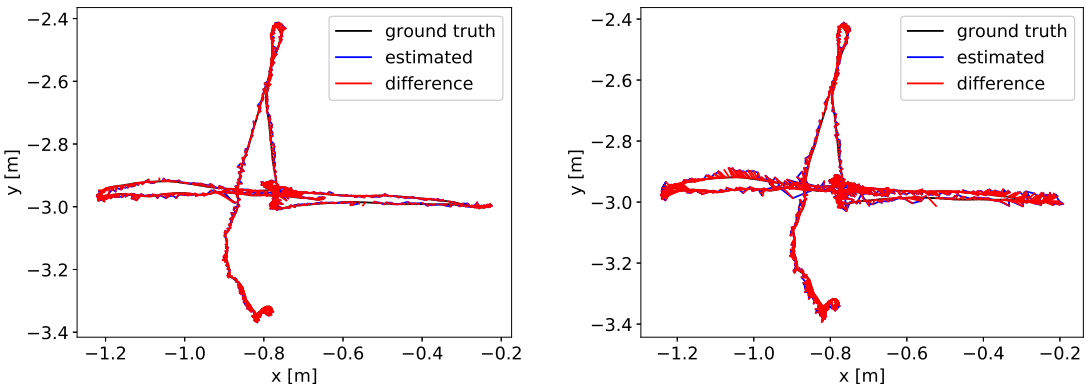
onde E é a trajetória estimada, G representa o *ground-truth*, e T é a transformação que alinha as duas trajetórias. Para uma sequência de N posições, o erro médio quadrático (RMSE) de ATE é dado por

$$RMSE(ATE_{1:N}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} ||trans(ATE_i)||^2} \tag{6}$$

Todos os testes foram realizados cinco vezes e as medianas dos resultados foram utilizadas para a avaliação, conforme proposto por [Mur-Artal et al. 2015].

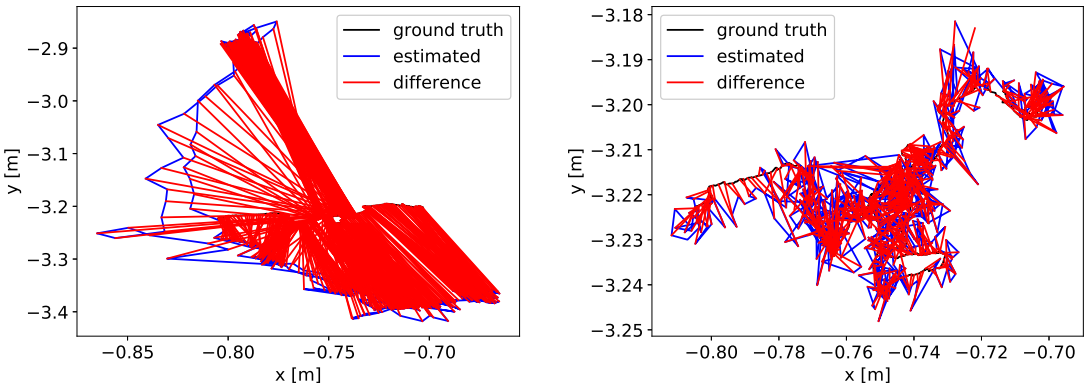
As Figuras 5, 6 e 7 mostram os gráficos de ATE de ORB-SLAM2 e do Crowd-SLAM para as sequências $fr3\_s\_xyz$, $fr3\_w\_static$ e $fr3\_w\_xyz$, respectivamente. Na sequência $fr3\_s\_xyz$, ambas trajetórias estão próximas da *ground-truth*. A natureza pouco dinâmica dessa sequência permite que o ORB-SLAM2 elimine os poucos pontos dinâmicos por meio de seus métodos de remoção de outliers, como o RANSAC. Nas sequências $fr3\_w$, por outro lado, o ORB-SLAM2 não é capaz de lidar com os pontos altamente dinâmicos e as trajetórias estimadas desviam da verdadeira. Já o sistema proposto

consegue uma trajetória bastante próxima da *ground-truth*.



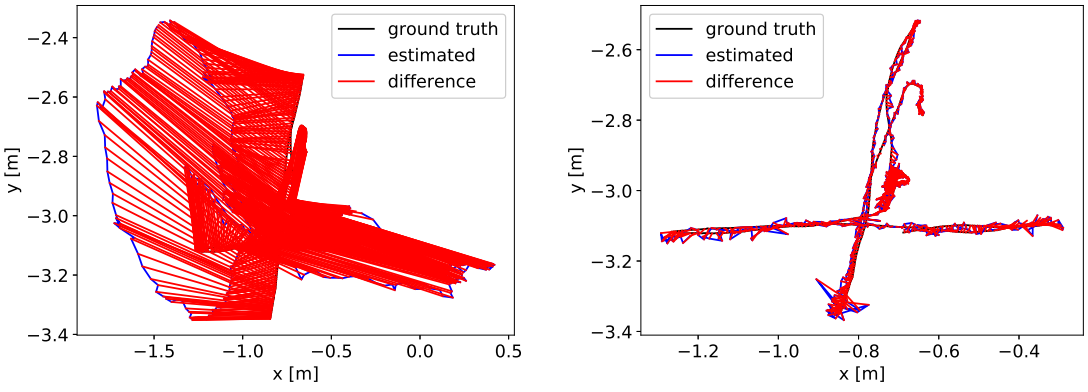
(a) ORB-SLAM2 **(b) Crowd-SLAM**

Figura 5. Ground-truth e trajetória estimada na sequência fr3_s_xyz



(a) ORB-SLAM2 **(b) Crowd-SLAM**

Figura 6. Ground-truth e trajetória estimada na sequência fr3_w_static



(a) ORB-SLAM2 **(b) Crowd-SLAM**

Figura 7. Ground-truth e trajetória estimada na sequência fr3_w_xyz

Além de ORB-SLAM2, o método proposto foi comparado com três métodos feitos para ambientes dinâmicos baseados em ORB-SLAM2: DS-SLAM [Yu et al. 2018],

DynaSLAM [Bescos et al. 2018] e SOF-SLAM [Cui and Ma 2019]. Os resultados são mostrados na Tabela 2. DynaSLAM tem os melhores resultados de RMSE em quatro sequências. No entanto, a diferença entre seus resultados e o sistema proposto está entre 1mm e 9mm, dependendo da sequência. Como o DynaSLAM é um método offline, nossos resultados são muito significativos.

Tabela 2. Comparação do RMSE (ATE) [m] do método proposto contra ORB-SLAM2, DS-SLAM, DynaSLAM e SOF-SLAM usando o dataset TUM

Sequência	ORB-SLAM2	DS-SLAM	DynaSLAM	SOF-SLAM	Crowd-SLAM
fr3_s_static	0.008	**0.006**	—	0.010	0.008
fr3_s_xyz	**0.009**	—	0.015	—	0.018
fr3_s_rpy	0.019	—	—	—	**0.015**
fr3_s_half	0.021	—	**0.017**	—	0.020
fr3_w_static	0.409	0.008	**0.006**	0.007	0.007
fr3_w_xyz	0.724	0.024	**0.015**	0.018	0.020
fr3_w_rpy	0.781	0.444	0.035	**0.027**	0.044
fr3_w_half	0.374	0.030	**0.025**	0.029	0.026

4.2. Detalhes de Implementação e Tempo de execução

Todos os testes foram realizados em um notebook com processador Intel Core i7 6700 HQ 2,60 GHz e 16 GB de RAM rodando Ubuntu Linux 18.04 LTS. O sistema é implementado em C++, e a detecção de objetos é realizada com OpenCV, utilizando apenas CPU. O trabalho proposto atingiu uma taxa média de 26,22 FPS, enquanto o ORB-SLAM2 atingiu 21,50 FPS. Para comparação, o Detect-SLAM, que também usa detecção de objetos, gasta 0,34 segundos por frame apenas para remoção de objetos em movimento. Dynamic-SLAM alcançou um desempenho médio de 22,2 FPS em fr3_w_xyz com GPU. Além disso, o desempenho alcançado pelo sistema proposto é maior do que a de outros sistemas que usam GPU, por exemplo, DynaSLAM (1,35 FPS) e DS-SLAM (13,08 FPS).

5. Conclusões

Este trabalho apresentou, de forma resumida, algumas contribuições desenvolvidas durante o doutorado, consistindo em um novo sistema de SLAM visual de código aberto projetado para funcionar em ambientes humanos lotados. O sistema é baseado no ORB-SLAM2, com quatro *threads* principais: rastreamento, detecção de objetos, mapeamento local e *loop closure*. Foi proposto um algoritmo eficiente de filtragem de *keypoints* dinâmicos, juntamente com uma rede neural treinada para detecção de objetos e um sistema de atualização do número de pontos detectados.

A eficácia do método proposto foi avaliada em sequências dinâmicas desafiadoras do *dataset* TUM, um dos mais usados na literatura de SLAM visual. Os resultados indicam que a metodologia proposta foi bem sucedida, com menor tempo computacional e melhor precisão em relação aos métodos do estado-da-arte.

Este trabalho é o primeiro método de SLAM visual feito especialmente para ambientes lotados, além de ser o primeiro método para ambientes dinâmicos de código aberto

que funciona em tempo real. Esse método facilita a viabilidade de navegação de robôs autônomos em ambientes externos e internos com muitas pessoas, como praças, fábricas, estações de trem ou hospitais, por exemplo. Além disso, é o método com melhor balanço entre velocidade de inferência e precisão da literatura, sem o uso de GPU.

Referências

Bescos, B., Fácil, J. M., Civera, J., and Neira, J. (2018). Dynaslam: Tracking, mapping, and inpainting in dynamic scenes. *IEEE Robotics and Automation Letters*, 3(4):4076–4083.

Cui, L. and Ma, C. (2019). Sof-slam: A semantic visual slam for dynamic environments. *IEEE Access*, 7:166528–166539.

Dendorfer, P., Rezatofighi, H., Milan, A., Shi, J., Cremers, D., Reid, I. D., Roth, S., Schindler, K., and Leal-Taix'e, L. (2020). Mot20: A benchmark for multi object tracking in crowded scenes. *ArXiv*, abs/2003.09003.

Endres, F., Hess, J., Sturm, J., Cremers, D., and Burgard, W. (2014). 3-d mapping with an rgb-d camera. *IEEE Transactions on Robotics*, 30:177 – 187.

Engel, J., Schöps, T., and Cremers, D. (2014). Lsd-slam: Large-scale direct monocular slam. In *Computer Vision – ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part II*, pages 834–849.

Galvez-López, D. and Tardos, J. D. (2012). Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969.

Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., and Burgard, W. (2011). g2o: A general framework for graph optimization. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*.

Labbé, M. and Michaud, F. (2019). Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, 36(2):416–446.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Liu, H., Liu, G., Tian, G., Xin, S., and Ji, Z. (2019). Visual slam based on dynamic object removal. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 596–601.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. (2016). Ssd: single shot multibox detector. In *Proceedings of the European conference on computer vision*, pages 21–37.

Milan, A., Leal-Taixé, L., Reid, I. D., Roth, S., and Schindler, K. (2016). Mot16: A benchmark for multi-object tracking. *ArXiv*, abs/1603.00831.

Mur-Artal, R., Montiel, J. M. M., and Tardós, J. D. (2015). Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163.

Mur-Artal, R. and Tardós, J. D. (2017). Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262.

Redmon, J. (2016). Darknet: Open source neural networks in c.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571.

Shao, S., Zhao, Z., Li, B., Xiao, T., Yu, G., Zhang, X., and Sun, J. (2018). Crowdhuman: A benchmark for detecting human in a crowd. *arXiv preprint arXiv:1805.00123*.

Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. (2012). A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580.

Xiao, L., Wang, J., Qiu, X., Rong, Z., and Zou, X. (2019). Dynamic-slam: Semantic monocular visual localization and mapping based on deep learning in dynamic environment. *Robotics and Autonomous Systems*, 117:1–16.

Yu, C., Liu, Z., Liu, X.-J., Xie, F., Yang, Y., Wei, Q., and Fei, Q. (2018). Ds-slam: A semantic visual slam towards dynamic environments. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1168–1174. IEEE.

Zhong, F., Wang, S., Zhang, Z., Chen, C., and Wang, Y. (2018). Detect-slam: Making object detection and slam mutually beneficial. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1001–1010.