ODOMETRIA VISUAL PARA CÂMERAS RGB-D EM MOVIMENTO

AFONSO HENRIQUES FONTES, JOSÉ EVERARDO BESSA MAIA
MESTRADO ACADÊMICO EM CIÊNCIA DA COMPUTAÇÃO, UNIVERSIDADE ESTADUAL DO CEARÁ
AV. DR. SILAS MUNGUBA, 1700 - CAMPUS DO ITAPERI, FORTALEZA/CE
E-MAILS: AFONSOHFONTES@GMAIL.COM, JOSE.MAIA@UECE.BR

Abstract – Visual odometry is the process of estimating the position and orientation of a camera analyzing the images associated to it. This paper develops a quick and accurate approach to visual odometry of a RGB-D camera moving in a static environment. The algorithm uses SURF (speeded Up Robust Features), RANSAC (Random Sample Consensus) and Minimum Mean Square to estimate the rigid transformation of six parameters between successive frames of video. Data from a Kinect camera were used in the tests. The results show that this approach is feasible and promising, surpassing in performance the algorithms ICP (Interative Closest Point) and SfM (Structure from Motion) in tests with publicly available dataset.

Keywords – Visual odometry, RGB-D, Rigid transformation.

Resumo – Odometria visual é o processo de estimar a posição e a orientação de uma câmera analisando as imagens a ela associadas. Este trabalho desenvolve uma abordagem rápida e precisa de odometria visual para uma câmera RGB-D navegando em ambiente estático. O algoritmo utiliza SURF (*Speeded Up Robust Features*), RANSAC (*Random Sample Consensus*) e Mínimo Quadrado Médio para estimar a transformação rígida de seis parâmetros entre quadros sucessivos do vídeo. Dados de uma câmera Kinect foram utilizados nos testes. Os resultados mostram que esta abordagem é viável e promissora tendo superado em performance os algoritmos ICP (*Interative Closest Point*) e SfM (*Structure from Motion*) em testes realizados com dados publicamente disponíveis.

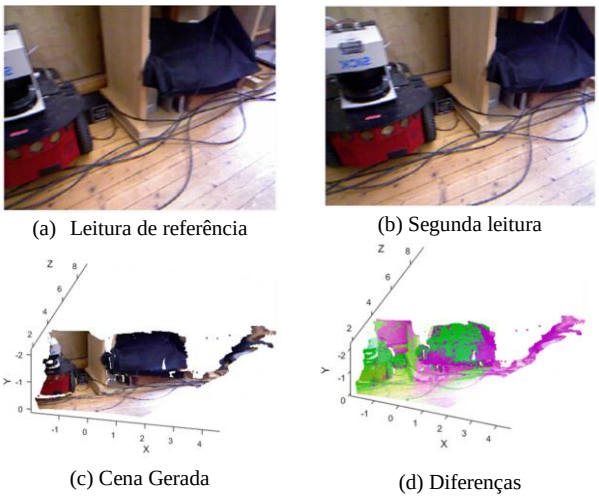
Palavras-chave – Visual odometry, RGB-D, Rigid transformation.

1 Introdução

Odometria visual, a capacidade de rastrear a postura (posição e orientação) ao longo do tempo utilizando somente informações visuais, tem sido bastante utilizada para a execução de tarefas de precisão em muitas aplicações relevantes nas áreas de robótica e visão computacional, especialmente quando nenhum outro sistema de sensoriamento está disponível (Konolige, 2008) e (Engel, 2012).

A contribuição deste artigo é prover uma odometria precisa e rápida utilizando informações obtidas através de um único sensor RGB-D: a câmera Microsoft Kinect. Este sensor captura tanto imagens coloridas como mapas densos de profundidade em taxas de quadros de vídeo. Ao combinar os pontos fortes de recursos visuais com informações de profundidade fornecidas pelo sensor, pode-se estimar com maior precisão a diferença de postura entre duas capturas. Isto permite, por exemplo, criar representações 3D densas de ambientes, como ilustra a Figura 1 com a captura de um par de quadros (*frames*) e a geração de uma cena através do algoritmo proposto.

Este trabalho está organizado entre três principais seções. A Seção 2 faz uma breve revisão de outras abordagens relacionadas à odometria visual e a Seção 3, apresenta o método proposto para odometria visual RGB-D e seus componentes. Uma análise de seu desempenho, na Seção 4, é feita utilizando um conjunto de dados (*dataset*) de referência que disponibiliza um grande número de sequencias de capturas obtidas através do Kinect. A trajetória real feita pela câmera RGB-D é obtida a partir de um sistema externo de captura de movimento de alta precisão. Desta forma o desempenho do algoritmo proposto pode ser comparado com outras contribuições.



(a) Leitura de referência (b) Segunda leitura

(c) Cena Gerada (d) Diferenças

Figura 1: A abordagem proposta estima a postura da câmera RGB-D entre duas capturas (a) e (b) ao calcular o movimento de corpo rígido entre elas. Com as informações de cor, profundidade e a postura estimada, podemos gerar uma cena em 3D (c), resultante da concatenação das profundidades verdes referentes a (a) e magenta a (b) que podem ser visualizados em (d).

2 Trabalhos Relacionados

Métodos de odometria visual baseados somente em imagens RGB, como a técnica de estrutura por movimento, ou *Structure from Motion* (SfM), executada por Benseddik (Benseddik, 2014), normalmente identificam pontos com características fortes em uma imagem de referência e estima suas localizações em

uma imagem subsequente para então calcular a transformação rígida entre os pontos. Um dos métodos mais utilizados na etapa de identificação de pontos relevantes é o SURF, siglas para *Speeded Up Robust Features* (Bay, 2008). Com o intuito de impedir falsas correspondências, se faz necessário filtrar aqueles pontos cuja variação de localização nas imagens não seguiu um padrão, geralmente utiliza-se amostra aleatória consensual, no inglês *Random Sample Consensus* (RANSAC), proposta por (Fischler, 1981), que filtra estes pontos inconsistentes através da estimação de parâmetros por amostragem. Em paralelo, e também com a finalidade de estimar a variação de postura entre quadros, existem métodos capazes de realizar a tarefa através da localização densa de pontos no espaço, como o *Interative Closest Point* (ICP) utilizado nos trabalhos de (Chetverikov, 2002) e (Milella, 2006) que minimiza a diferença entre duas nuvens de pontos, sendo irrelevante a cor (RGB) dos mesmos.

Para cada quadro capturado, uma câmera RGB-D, como a utilizada neste trabalho, nos fornece uma imagem colorida (RGB) e uma imagem de profundidade (*Depth*). Isto nos possibilita trabalhar com todas as técnicas citadas anteriormente separadamente e/ou mesclá-las, criando uma nova abordagem, como fazem as contribuições de (Konolige, 2008) e (Engel, 2012).

Finalmente, o método contemplado neste artigo consiste em três principais etapas de processamento, como ilustrado na Figura 2. Primeiramente é executado SURF em um par de imagens RGB para identificar regiões com características relevantes, extraindo pontos de interesse entre as capturas, depois filtra-se aqueles pares pontos que não seguem um padrão de deslocamento utilizando RANSAC, como nos trabalhos de Benseddik (Benseddik, 2014) e Kneip (Kneip, 2011), sendo diferente destes na terceira etapa quando a profundidade dos pontos é obtida utilizando as imagens *Depth*, e não estimada por triangulação como os trabalhos citados. Uma vez obtido um conjunto de pontos no espaço referentes à primeira captura e as suas localizações em uma captura subsequente, a transformação rígida entre as leituras, representada por rotação e translação, pode ser calculada utilizando decomposição em valores singulares ou *Single Value Decomposition* (SVD), cujo passo a passo está documentado na próxima seção deste artigo. Pode-se associar a transformação rígida computada à diferença entre as posturas da câmera nas posições atual e anterior, *i* e *i*-1. Obviamente, para estimar a postura atual da câmera, é preciso considerar todas as transformações calculadas anteriormente.

Outro algoritmo que se assemelha com o proposto neste artigo é o do trabalho de Steinbrücker (Steinbrücker, 2011), com o diferencial de que o autor trabalha de forma densa, levando em consideração todos os pontos lidos, atribuindo pesos a cada um deles, já o método aqui descrito não atribui pesos e trabalha de forma esparsa, levando em consideração apenas os pontos fortes de cada quadro. Como resultado o algoritmo fornece uma sequência de posturas associadas a cada par de capturas, as quais podem ser utilizadas em diversas aplicações, como reconstrução de cenas em 3D (Fernández-Moral, 2016), sistemas de localização e mapeamento simultâneos (SLAM) (Gao, 2015), ou até para fins médicos, como feito por Spyrou (Spyrou, 2015), que utiliza odometria visual em imagens capturadas durante um exame de endoscopia.

3 Proposta

Esta seção apresenta a abordagem utilizada neste trabalho para estimar o movimento rígido feito pela câmera entre um par de imagens RGB-D. A abordagem é uma versão baseada nos trabalhos recentes de Benseddik (Benseddik, 2014) e (Sorkine, 2009) e está dividida entre duas subseções, sendo a primeira referente à identificação e filtro de pontos relevantes nas imagens RGB e a segunda ao cálculo da postura, divisão ilustrada respectivamente entre os pontos um e dois da Figura 2.

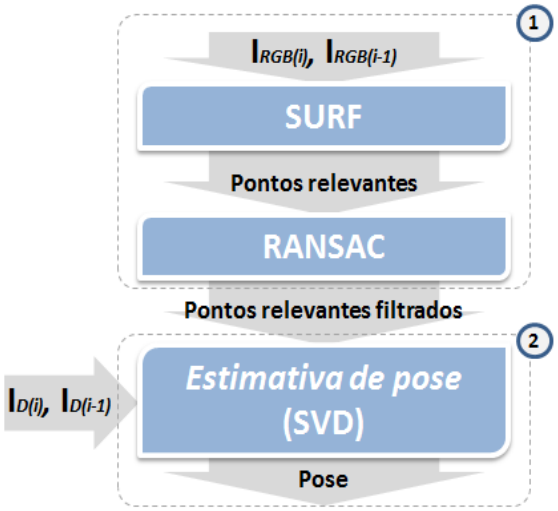


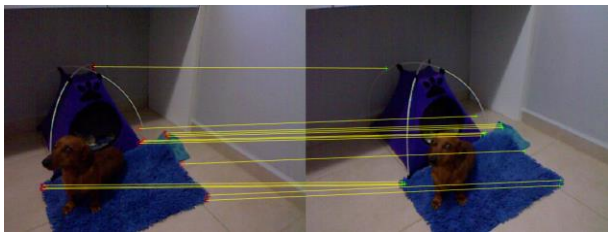
Figura 2. Fluxograma de técnicas aplicadas para estimar a postura da câmera a partir de um par de imagens RGB, $I_{RGB(i)}$ e $I_{RGB(i-1)}$, e suas respectivas imagens de profundidade, $I_{D(i)}$ e $I_{D(i-1)}$. Separado em duas principais etapas: extração de pontos (1) e estimativa de pose (2).

2.1 Encontrando pontos relevantes

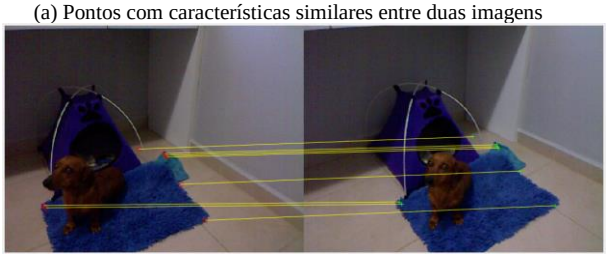
SURF é uma técnica robusta para detecção de pontos chave em imagens, principalmente quando as mesmas estão sujeitas a variação como rotação, deslocamento e mudança de escala. Apresentado pela primeira vez por Herbert Bay em 2008 (Bay, 2008), SURF é frequentemente utilizado na visão computacional em tarefas como reconhecimento de objetos ou reconstrução 3D (Henry, 2014). A Figura 3 ilustra duas imagens e os pontos com características similares em ambas, note que existem pontos que não seguem o padrão de deslocamento entre as capturas em (a), sendo assim preciso a utilização de um filtro antes do cálculo da postura.

O algoritmo RANSAC, proposto por Fischler (Fischler ,1981), trabalha com a hipótese de que o conjunto de dados em questão possui *inliers* e *outliers*, ou seja, que podem ser divididos entre valores dentro de uma faixa de aceitação e valores discrepantes (Henry, 2014). No contexto deste trabalho, o algoritmo RANSAC é utilizado para filtrar conjuntos de pontos (coordenadas) que não seguem um padrão de deslocamento entre as imagens, como nos mostra a Figura 3, o que permite trabalhar somente com pontos que provavelmente possuam a mesma rotação e translação, assim o cálculo de estimativa da postura tende a ser mais preciso. Em resumo, ao remover falsas correspondências, a precisão aumenta.

Uma vez encontradas e filtradas as coordenadas de interesse, *x* e *y*, dos pontos relevantes, se faz necessário saber suas respectivas profundidades *z*, antes de prosseguir com o calculo da postura. Isto é relativamente simples, já que a câmera RGB-D fornece informações de profundidade através das imagens *Depth*. A Figura 4 ilustra em 3 dimensões as profundidades referentes a primeira captura da Figura 3.



(a) Pontos com características similares entre duas imagens



(b) Pontos remanescentes após o filtro

Figura 3. Utilização da técnica SURF em um par de imagens (a) e em seguida RANSAC (b).

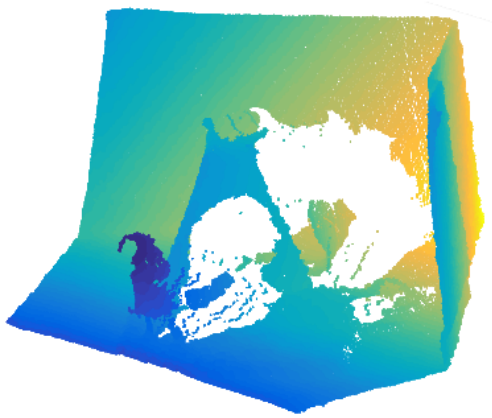


Figura 4. Informação de profundidade de um quadro (*frame*) representada em 3D.

2.2 Estimando a Postura

Dado um par de capturas, pode-se associar a diferença entre as posturas atual e anterior ao calculo da transformação rígida entre os *frames*. Obviamente, para estimar a postura atual da câmera, devemos considerar todas as transformações calculadas anteriormente.

McCarthy (McCarthy, 1990) explica que é possível representar uma transformação rígida por um vetor de translação ***t*** e uma matriz rotação ***R*** de tamanhos 1 x *d* e *d* x *d*, respectivamente, onde *d* é a dimensão trabalhada, ou seja, assume o valor 2 para 2D e 3 para 3D (McCarthy, 1990). Neste trabalho, ***t*** é igual às coordenadas cartesianas *x*, *y* e *z*, nesta ordem, enquanto a matriz identidade 3x3 representa uma rotação ***R*** nula em todos os eixos. Segundo McCathy, as rotações ao longo de cada eixo é dada por (1), (2) e (3).

eixo x,

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(a_x) & sen(a_x) \\ 0 & -sen(a_x) & \cos(a_x) \end{pmatrix} \quad (1)$$

eixo y,

$$\begin{pmatrix} \cos(a_y) & 0 & -sen(a_y) \\ 0 & 1 & 0 \\ sen(a_y) & 0 & \cos(a_y) \end{pmatrix} \quad (2)$$

eixo z,

$$\begin{pmatrix} \cos(a_z) & sen(a_z) & 0 \\ -sen(a_z) & \cos(a_z) & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (3)$$

Sejam então ***P*** = $\{p_1, p_2, ..., p_n\}$ e ***Q*** = $\{q_1, q_2, ..., q_n\}$ dois conjuntos de *n* coordenadas *x*, *y* e *z*, o objetivo desta etapa é encontrar uma rotação ***R*** e uma translação ***t*** que minimize o alinhamento entre ***P*** e ***Q***, conforme expressa a equação (4), (Sorkine, 2009).

$$(R,t) = \min \sum_{i=1}^{n} |(R \times p_i + t) - q_i|^2 \quad (4)$$

Sorkine propõe cinco passos para calcularmos ***R*** e ***t***, são eles:

- Computar a centroide dos conjuntos de pontos ***P*** e ***Q*** (5).

$$\bar{p} = \frac{\sum_{i=1}^{n} p_i}{n} \ , \ \bar{q} = \frac{\sum_{i=1}^{n} q_i}{n} \quad (5)$$

- Calcular os vetores centralizados (6).

$$v1_i := p_i - \bar{p} \ , \ v2_i := q_i - \bar{q} \ , \quad i = 1, 2, ..., n \quad (6)$$

- Encontrar a matriz de covariância *d* x *d* (7), onde, neste caso, *X* e *Y* são as matrizes *d* x *n* que possuem respectivamente ***v1****i* e ***v2****i* como colunas, e *I* sendo a matriz identidade *n* x *n*.

$$S = X \times I \times Y^T \quad (7)$$

- Computar a decomposição em valores singulares, dada por $S = U\Sigma V^T$, e finalmente calcular a rotação utilizando (8).

$$R = V \begin{pmatrix} 1 & & \\ & \ldots & \\ & & \det(VU^T) \end{pmatrix} U^T \quad (8)$$

- Por último, o calculo da translação está descrito em (9).

$$t = \bar{q} - R\bar{p} \quad (9)$$

3 Resultados experimentais

Para avaliar qualitativamente o algoritmo proposto neste trabalho, foram utilizados os conjuntos de imagens RGB-D disponibilizadas ao público por Sturm (Sturm, 2012), as quais foram adquiridas a partir de uma câmera Microsoft Kinect idêntica a utilizada neste artigo. Juntamente com as imagens, Sturm também disponibiliza a trajetória real feita pela câmera (*groundtruth*), esta adquirida por um sistema externo de captura de movimento de alta precisão. Há uma grande variedade de capturas disponíveis para *download*, porém, a avaliação de desempenho foi feita utilizando as sequências denominadas freiburg1/floor e freiburg1/teddy pois contêm tanto movimento de translação quanto de rotação em um ambiente típico de escritório em velocidades e percursos diferentes. A câmera RGB-D foi movimentada manualmente em ambas as sequências com o diferencial de que em freiburg1/floor a mesma não perde a referencia do chão em maioria dos seus quadros (*frames*).

Os experimentos foram realizados em MATLAB versão R2016a instalado em um sistema operacional Windows 10 x64 sendo executado em um *Laptop* com processador Intel Core I3 e 4GB de memória RAM. Para obtenção dos resultados aqui expostos, também foi necessário a instalação da extensão para MATLAB *Computer Vision System toolbox*. Os ensaios consistem em percorrer todos os *frames* estimando a variação de postura da câmera entre eles, levando em consideração somente as leituras atual e anterior. Para comparação de desempenho, além da execução do algoritmo descrito na segunda seção deste artigo, computamos a variação de postura utilizando duas técnicas distintas: *Interative Closest Points* (ICP), que leva em consideração apenas imagens de profundidade (*Depth*), e a técnica de estrutura por movimento, ou *Structure from Motion* (SfM), executada por Benseddik (Benseddik, 2014), que segue um fluxo similar ao ilustrado na Figura 2, com o diferencial de que a estimativa de profundidade é feita por triangulação, sendo necessárias apenas imagens coloridas (RGB) para estimar a postura. Desta forma pretende-se comparar distintas abordagens de Odometria Visual.

A partir da trajetória real da câmera RGB-D disponibilizada nos *datasets* de Sturm, é possível gerar duas visões do erro para cada experimento: *frame* a *frame*, Figuras 5 e 7, e acumulado, Figuras 6 e 8. A visão acumulada computa a diferença entre a posição real da câmera no ambiente e a posição estimada, estas sendo resultado da concatenação de todas as movimentações anteriores e impressas na visão *frame* a *frame*. Note que, mesmo que a postura da câmera seja composta por três coordenadas de localização e respectivamente três ângulos de orientação, em ambas as visões, o erro, impresso em metros, é calculado levando em consideração somente a localização da câmera no ambiente, pois esta é diretamente afetada por sua orientação.

No primeiro experimento, freiburg1/floor, ilustrado nas Figuras 5 e 6, a câmera navega por um ambiente de escritório tendo o chão como referência em maioria de suas capturas. É possível notar na Figura 5 que o algoritmo ICP, representado pela legenda *Depth*, possui os picos mais significativos, seguido pelo método proposto nesse artigo, sendo SfM o método com maior constância de pequenos erros de posição. Porém, o erro de orientação, que pode ser melhor visualizado no gráfico acumulativo da Figura 6, inverte a primeira impressão ao ilustrar uma diferença de posição entre SfM e a real superior a 1,5 metros ao serem percorridos 900 *frames*.

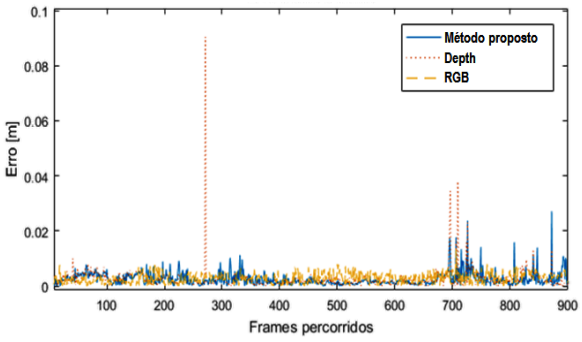


Figura 5. Erro computado *frame* a *frame* comparando o método proposto neste artigo com os algoritmos ICP e SfM, representados respectivamente pelas legendas *Depth* e RGB, para o conjunto de dados freiburg1/floor.

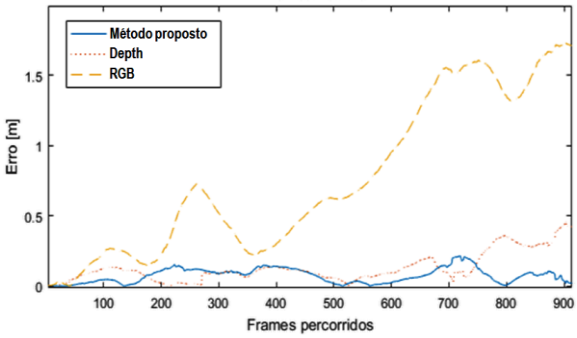


Figura 6. Erro acumulado comparando o método proposto neste artigo com os algoritmos ICP e SfM, representados respectivamente pelas legendas *Depth* e RGB, para o conjunto de dados freiburg1/floor.

Já no segundo experimento, freiburg1/teddy, ilustrado pelas Figuras 7 e 8, a câmera realiza movimentos de orientação mais bruscos e a sequência de capturas não tem nenhuma referência constante, como o chão ou o teto. Nele é possível notar um melhor desempenho do algoritmo SfM, que novamente apresenta a melhor constância de pequenos erros *frame* a *frame*, visualizado na Figura 7, seguido pelo método proposto e por último pelo algoritmo ICP, e, além disso, possui o menor erro acumulado ao serem atingidos 1000 *frames*.

Os erros médios dos três métodos para os experimentos realizados podem ser visualizados nas Tabelas 1 e 2. Nelas é possível notar que o método proposto possui os menores erros médios e um tempo de processamento relativamente baixo em ambos os experimentos.

Durante a execução dos experimentos foi possível notar que o algoritmo ICP é imune a variação de cor ou variação de luz no ambiente navegado, entretanto falha com maior frequência quando o sensor não detecta grande variação de profundidade. Enquanto o algoritmo SfM depende fortemente da variação de cor entre uma imagem e outra para melhorar sua precisão, sendo incapaz de navegar em ambientes escuros. Esta diferença entre os dois algoritmos pode ser observada nos trechos entre os *frames* 550 e 650,e 950 e 1050 da Figura 8, quando a câmera navega em uma área com muita variação de profundidade e muita variação de cor respectivamente.

Os resultados dos experimentos mostram que o método proposto neste artigo possui os menores erros médios, um tempo de processamento relativamente baixo e uma constância satisfatória quando comparado aos algoritmos citados.

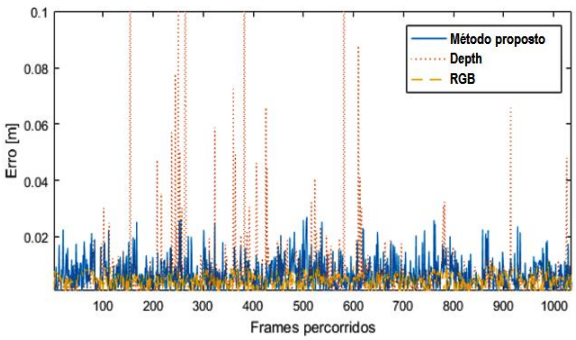


Figura 7. Erro computado *frame* a *frame* comparando o método proposto neste artigo com os algoritmos ICP e SfM, representados respectivamente pelas legendas *Depth* e RGB, para o conjunto de dados freiburg1/teddy.

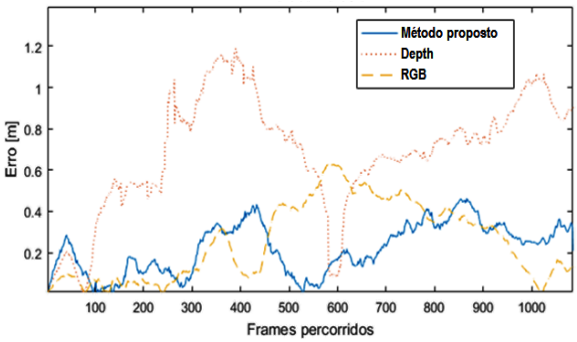


Figura 8. Erro acumulado comparando o método proposto neste artigo com os algoritmos ICP e SfM, representados respectivamente pelas legendas *Depth* e RGB, para o conjunto de dados freiburg1/teddy.

Tabela 1. Erro acumulado médio e tempo de processamento para o experimento freiburg1/floor.

Algoritmo	Erro médio	Tempo
SfM	0,74 m	950 s
ICP	0,12 m	621 s
Método Proposto	0,09 m	658 s

Tabela 2. Erro acumulado médio e tempo de processamento para o experimento freiburg1/teddy.

Algoritmo	Erro médio	Tempo
SfM	0,2689 m	2226 s
ICP	0,6629 m	1638 s
Método Proposto	0,2633 m	1388 s

5 Conclusão

Este artigo apresentou uma abordagem de Odometria Visual para câmeras RGB-D navegando em um ambiente estático. O método proposto é comparado com SfM e ICP, frequentemente utilizados em diversas aplicações. O desempenho foi avaliado em precisão e tempo de processamento. Para as duas sequencias de capturas utilizadas, o método proposto obteve uma precisão em média 73% melhor que ICP e 46% melhor que SfM e um tempo de processamento em média 12% melhor que ICP e 53% melhor que SfM.

O algoritmo proposto trabalha somente com pares de capturas, ignorando informações já processadas, o que limita seu uso a ambientes estáticos e, caso utilizado para reconstruir ambientes em 3D, pode não pode não resultar no posicionamento ideal do frame atual no que diz respeito ao contexto global da reconstrução, gerando sobreposição de informação. Pode-se melhorar seu desempenho com o uso de algoritmos de otimização como TORO (Grisetti, 2009) e g2o (Kümmerle, 2011).

Referências Bibliográficas

Konolige, K et al, 2008. Outdoor mapping and navigation using stereo vision. Experimental Robotics. Springer Berlin Heidelberg. p. 179-190.

Engel, J; Sturm, J; Cremers, D, 2012. Camera-based navigation of a low-cost quadrocopter. Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on. IEEE. p. 2815-2821.

Bensedikk, H; Djekoune, O; Blhocine, M, 2014. Sift and surf performance evaluation for mobile robot-monocular visual odometry.Journal of Image and Graphics, v. 2, n. 1, p. 70-76.

Sturm, J et al, 2012. A benchmark for the evaluation of RGB-D SLAM systems. Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on. IEEE. p. 573-580.

Bay, Herbert et al. Speeded-up robust features (SURF). Computer vision and image understanding, v. 110, n. 3, p. 346-359, 2008.

Fischler, M.; Bollers, R, 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, v. 24, n. 6, p. 381-395.

Chetverikov, D et al, 2002. The trimmed iterative closest point algorithm. In: Pattern Recognition, 2002. Proceedings. 16th International Conference on. IEEE. p. 545-548.

Mirella, A; Siegwart, R, 2006. Stereo-based ego-motion estimation using pixel tracking and iterative closest point. In: Computer Vision Systems, 2006 ICVS'06. IEEE International Conference on. IEEE. p. 21-21.

Kneip, L; Chli, M; Siegwart, R, 2011. Robust real-time visual odometry with a single camera and an imu. In: BMVC. p. 1-11.

Steinbücker, F; Sturm, J; Cremers, D, 2011. Real-time visual odometry from dense RGB-D images. In: Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on. IEEE. p. 719-722.

Fernández-Moral, E et al, 2016. Scene structure registration for localization and mapping. Robotics and Autonomous Systems, v. 75, p. 649-660.

Gao, X; Zhang, T, 2015. Robust RGB-D simultaneous localization and mapping using planar point features. Robotics and Autonomous Systems, v. 72, p. 1-14.

Spyrou, E et al, 2015. Comparative assessment of feature extraction methods for visual odometry in wireless capsule endoscopy. Computers in biology and medicine, v. 65, p. 297-307.

Sorkine, O, 2009. Least-squares rigid motion using svd. Technical notes, v. 120, p. 3.

McCarthy, J, 1990. Introduction to theoretical kinematics. MIT press.

Henry, P et al, 2014. RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In: Experimental robotics. Springer Berlin Heidelberg. p. 477-491.

Grisetti, G. et al, 2009. Toro-tree-based network optimizer. 2014-03-15]. http://www. openslam. org/toro. html.

Kümmerle, R et al, 2011. g 2 o: A general framework for graph optimization. In: Robotics and Automation (ICRA), 2011 IEEE International Conference on. IEEE. p. 3607-3613.