Análise de Desempenho de Algoritmos Detectores de *Keypoints* para um Sistema de Navegação Visual de Robôs Baseados em Smartphones

Bruno A. S. Santana, Rosiery da S. Maia, Wilfredo B. Figuerola, Anderson A. S. Souza
Departamento de Informática
Universidade do Estado do Rio Grande do Norte
Natal, Brasil
Email: bruno.basswn@hotmail.com, rosiery@uern.br, wblancof@gmail.com, andersonabner@uern.br

***Resumo*—Os *smartphones* despontaram em pesquisas recentes, como uma possível plataforma alternativa para a aquisição, processamento e controle de sistemas robóticos móveis (os chamados CellBots). A possibilidade de ser ter um *smartphone* como centro de comando de um robô móvel se tornou atrativa, pois em um único dispositivo é possível coletar dados de câmeras, GPS, acelerômetro, entre outros sensores, além das capacidades de comunicação desses aparelhos. Através da aquisição e processamento das imagens das câmeras de um *smartphone*, por exemplo, é possível coletar dados úteis para a implementação de um sistema de navegação robótico puramente visual com: odometria visual, mapeamento visual e reconhecimento de objetos. Este trabalho apresenta um estudo de três dos principais algoritmos de detecção de características (pontos notáveis) para a implementação de um sistema de navegação visual, voltado para um CellBot. Um estudo comparativo feito a partir da implementação e testes desses algoritmos em um *smartphone* é apresentado no final deste trabalho, a fim de facilitar a escolha do algoritmo mais adequado a ser utilizado na implementação do sistema de navegação.**

***Keywords*-Visão Computacional; Sistema de Navegação Visual; CellBots.**

***Abstract*—Smartphones have emerged in recent research, as a possible alternative platform for acquisition, processing and control of mobile robotic systems (Cellbots). Having a smartphone as a mobile robot command center became attractive because it can collect data from cameras, GPS, accelerometer and other sensors, besides their communications capabilities. By acquiring and processing the images from the smartphone cameras, for example, it is possible to collect useful data for the implementation of a robotic visual navigation system with visual odometry, visual mapping and object recognition. This paper presents a study of three major keypoints detection algorithms to implement a visual navigation system for a CellBot. A comparative study from the implementations and testing of these algorithms on a smartphone is presented at the end of this work, in order to facilitate the choice of the most appropriate algorithm to be used in the navigation system.**

***Keywords*-Computer Vision; Visual Navigation System; CellBots.**

I. Introdução

Um dos principais objetivos dos cientista que trabalham com a Robótica é fazer com que os robôs sejam autônomos, isto é, que não necessitem da interferência humana para realizar suas tarefas [1]. Para isso, é necessário que um robô tenha a capacidade de sentir, planejar e agir [2]. A capacidade de sentir está diretamente ligada aos sensores que um robô possui para perceber o seu ambiente. Assim, quanto mais sensores, originando informações úteis para o robô, maior será a capacidade de percepção do mesmo. Considerando os dados capturados e processados pelos sensores, o robô deve planejar ações, que o leve a cumprir com seu objetivo. Após a etapa de planejamento, o robô deve acionar seus atuadores de forma a interagir com o ambiente no qual está inserido, obedecendo as ações que foram planejadas para alcançar a completude da tarefa especificada.

Construir e/ou montar um robô com tais capacidades se torna algo oneroso, tanto no sentido monetário quanto no tempo gasto para desenvolvê-lo. Neste sentido, alguns pesquisadores destinam seus esforços para propor soluções rápidas e de baixo custo no desenvolvimento de robôs autônomos [3][4][5]. Neste contexto, surgiram os robôs baseados em *smartphones*, os chamados CellBots, que utilizam as funcionalidades dos *smartphones* para empregá-los como computador controlador, sem a necessidade de lidar circuitos eletrônicos complexos.

Este modelo permite que o robô aproveite as capacidades sensoriais e de processamento dos celulares mais modernos, os quais são frequentemente equipados com GPS (*Global Positioning System*), câmeras compassos de três eixos e acelerômetro, wifi, bluetooth, alto-falantes, microfone, entre outras características que se adequam a um sistema de controle robótico [6]. Além disso, já foi mostrado a viabilidade de se executar algoritmos de navegação complexos usando os processadores de *smartphones* [7]. Esta atraente possibilidade foi confirmada em uma reportagem na revista IEEE Robotics and Automation com o título "Smartphone-based robots: The new robot brains"como uma tendência na robótica [8].

Sabe-se também, que é possível utilizar dados visuais de câmeras para implementar sistemas de navegação robótica robustos. Esses sistemas geralmente são baseados em algoritmos de detecção de pontos notáveis (*keypoints*) encontrados no ambiente no qual o robô está inserido [9][10].

Diante deste contexto, este trabalho apresenta um estudo sobre a viabilidade de se utilizar três dos principais algoritmos de detecção de pontos notáveis (*features*), para a utilização dos mesmos na implementação de um sistema de navegação visual para um robô baseado em *smartphone*. Os algoritmos estudados são: o SIFT (*Scale-Ivariant Feature Transform*), SURF (*Speeded Up Robust Features*) e o ORB (*Oriented FAST and Rotated BRIEF*). Os mesmos foram implementados em um *smartphone* com o sistema operacional Android munido da biblioteca de visão computacional OpenCV (*Open Source Computer Vision*). Análises de desempenho desses algoritmos foram feitas, levando-se em consideração várias amostras de imagens. No final deste estudo, pretende-se apontar o algoritmo que mais se adequa ao sistema de navegação de um CellBot.

Este trabalho segue com a Seção II, que traz a fundamnetação teórica para se entender os algoritmos estudados; as Seções III e IV mostram os experimentos e as análises feitas com os dados obtidos, apontando o algoritmo de melhor desempenho; e a Seção V traz os aspectos conclusivos deste estudo realizado.

II. Fundamentação Teórica

A. Detecção de Características

Em Visão Computacional o termo característica se refere a uma parte de uma imagem com alguma propriedade especial, por exemplo, um círculo, uma linha, ou uma região textura, pontos, curvas e superfícies [11]. Essas características podem ser utilizadas para diferentes propósitos: reconhecimento de objetos, odometria visual, mapeamento, detecção de movimentos, entre outros.

Porém, são necessários algoritmos de detecção para que essas características possam ser extraídas e utilizadas. Esses algoritmos produzem como saídas um conjunto de descritores (*features descriptors*), os quais especificam a posição e outras propriedades essenciais das características encontradas na imagem. Os algoritmos que possuem essa capacidade de detecção e descrição de características em uma imagem também são chamados de *extratores de características* (*features extractors*) [12].

Algoritmos extratores de pontos (ou pontos notáveis, ou pontos de interesse, ou *keypoints*) estão entre os extratores de características mais utilizados, quando o propósito de uso está relacionado com alguma atividade da robótica. Entre eles se destacam três algoritmos considerados robustos e eficientes: o SIFT, o SURF e o ORB [13][14][15].

Esses algoritmos suportam variações tanto de escala quanto de rotação, ou seja, mesmo que uma imagem seja redimensionada, ou mesmo que seja rotacionada, esses algoritmos são capazes de detectar os pontos notáveis. Essa é uma das principais vantagens de se utilizá-los nas aplicações relacionadas à robótica, visto que os robôs estão em constante movimento.

B. SIFT - Scale-Invariant Feature Transform

O algoritmo SIFT foi proposto por [13], com a divisão em etapas de execução. A primeira etapa do algoritmo SIFT consiste em identificar as localizações e escalas repetitivamente assinaladas de um ponto. A detecção das localizações que são invariantes à mudanças de escala da imagem pode ser alcançada através da busca por características estáveis em todas as possíveis escalas dentro de um espaço de escalas [13].

O processo se inicia através da geração progressiva de cinco imagens, pela aplicação de um filtro gaussiano (cinco níveis de desfoque). Essas imagens geradas passam a formar uma *octave* (ver Figura 1). Após, a imagem original é redimensionada pela metade do tamanho original e gera-se uma nova *octave*, aplicando-se a operação de filtragem gaussiana. Esse processo se repete até que quatro *octaves* sejam criadas, concretizando a formação do espaço de escalas.

No passo anterior, foi criado o espaço de escala, cujas imagens são utilizadas para gerar um outro conjunto, as imagens da Diferença do Gaussiano (*Difference of Gaussian - DoG*). O filtro DoG é uma aproximação do Laplaciano do Gaussiano (*Laplacian of Gaussian - LoG*), e o LoG é um filtro que, ao ser aplicado em uma imagem, facilita o encontro dos pontos notáveis. A criação das imagens da DoG se dá da seguinte forma: imagens consecutivas em uma *octave* são subtraídas para produzir uma imagem da DoG. Então o próximo par consecutivo é tomado, e o processo se repete. Isso é feito para todas as *octaves*. A Figura 1 ilustra os dois primeiros passos do algoritmo SIFT.

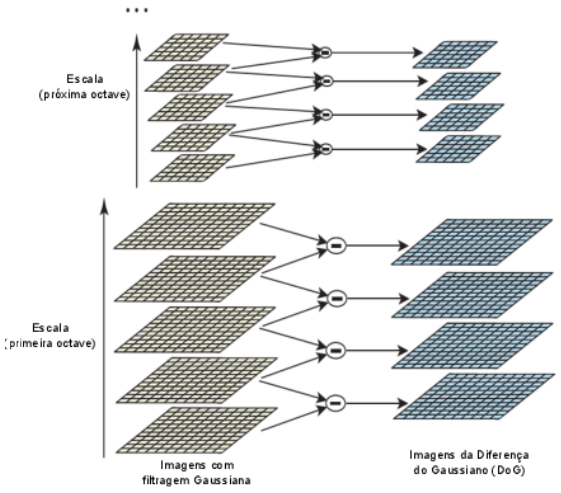


Figura 1. Variação de escala em uma imagem.

Neste ponto, se inicializa a busca pelos pontos notáveis. Localiza-se, de forma grosseira, os máximos e mínimos

das imagens. Para cada pixel de uma imagem da DoG, faz-se a comparação dele com seus vizinhos, tanto na imagem atual, quanto nas imagens acima e abaixo, dentro do espaço de escala. Se o pixel examinado é um máximo ou mínimo, então ele é um potencial candidato a ser um ponto notável.

Uma vez que as localizações dos potenciais pontos notáveis são encontradas, faz-se um refinamento desses valores através da aplicação da Série de Taylor para obter-se resultados mais acurados. Após, elimina-se os pixels cujo o valor de intensidade é menor que um limiar estabelecido ou se eles estão localizados sobre uma aresta.

Neste ponto, tem-se os pontos notáveis detectados, mesmo com a variação na escala das imagens. O próximo passo é assinalar a orientação de cada *keypoint*. A orientação provê a invariância na rotação, ou seja, mesmo que a imagem seja rotacionada, os *keypoints* serão identificados. A ideia é computar a direção do gradiente e sua magnitude da vizinhança de cada pixel. Usando um histograma de orientações, a orientação do gradiente mais proeminente é identificada. O pico de orientação identificado no histograma, é atribuído ao ponto notável em questão. Caso exista mais de um pico acima de um valor especificado, ele também será utilizado no cálculo da orientação.

O passo final do algoritmo consiste na criação do descritor. Uma janela de 16×16 pixels é criada na vizinhança de um ponto notável. Ela é dividia em dezesseis sub-blocos de 4×4 pixels. Dentro de cada sub-bloco, as magnitudes do gradiente e as orientações são calculadas (ver Figura 2).

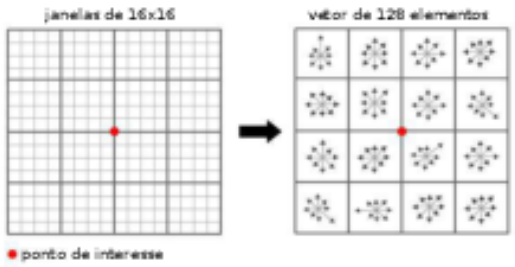


Figura 2. Criação do descritor SIFT.

As orientações são colocadas em um histograma de oito divisões, assim, um total de 128 intervalos de valores são disponibilizados. Uma vez que se tem os 128 valores, normaliza-se todos, resultando no descritor do ponto notável detectado, também chamado de (*feature vector*).

C. SURF - Speed-Up Robust Features

O algoritmo SURF é, na verdade, uma versão mais acelerada do SIFT. Para gerar o espaço de escalas o SIFT faz uma aproximação do Laplaciano do Gaussiano com a Diferença do Gaussiano. O SURF vai um pouco mais além, e aproxima o LoG com um filtro retangular *Box Filter*. Uma vantagem dessa aproximação é que a operação de convolução com o filtro retangular pode ser facilmente calculada com a ajuda de imagens integral, estruturas de dados utilizadas para a realização mais rápida de operações como média e somas em uma região de uma imagem. Essa operação de convolução pode ser feita em paralelo para diferentes escalas.

Para assinalar a orientação de um ponto de interesse, o SURF usa respostas da aplicação da Transformada de Haar (*Haar-wavelet*) nas direções x e y, dentro de uma região de vizinhança especificada. Uma vez que as respostas à wavelet são calculadas e ponderadas por uma Gaussiana centrada no ponto de interesse, as respostas são plotadas em um dado espaço (ver Figura 3). A orientação dominante é estimada calculando-se a soma de todas as respostas que estão dentro de uma área do espaço com ângulo de 60 graus. Essas respostas à wavelet podem ser, também, facilmente calculadas usando-se as imagens integral, usadas anteriormente na criação do espaço de escalas.

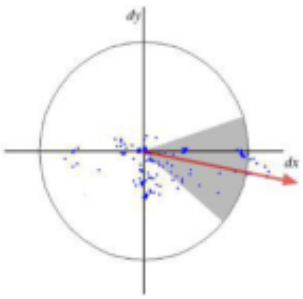


Figura 3. Variação de escala em uma imagem.

Para a composição do descritor, o SURF define uma vizinhança em torno do ponto em questão. Essa janela é subdividida em regiões de 4×4. Para cada elemento da sub região, as respostas da Haar-wavelet nas direções de x e y são tomadas, e um vetor é formado como de acordo com a expressão 1.

$$v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|) \qquad (1)$$

Onde d_x e d_y são as respostas da Haar-wavelet nas direções x e y respectivamente. Dessa forma, cada sub região de 4×4 é representada pela concatenação de quatro vetores, formando um vetor de 16 elementos. No final, o descritor será formado pela concatenação dos vetores de todas as sub regiões, formando um vetor descritos de 64 elementos. A Figura 4 mostra o processo de formação do descritor SURF.

D. ORB - Oriented FAST and Rotated BRIEF

O ORB é um algoritmo que surgiu como uma opção alternativa ao SIFT e ao SURF, propondo melhor desempenho computacional [15]. O ORB é, basicamente, uma fusão do detector FAST (*Features from Accelerated Segment Test*) com o descritor BRIEF (*Binary Robust Independent Elementary Features*).

O algoritmo inicia com o FAST para encontrar pontos notáveis, e em seguida aplica um filtro de cantos de Harris

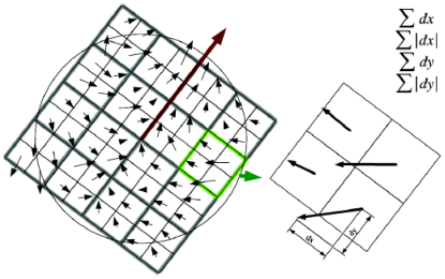


Figura 4. Variação de escala em uma imagem.

para separar os melhores pontos entre os encontrados. O algoritmo também utiliza um esquema de pirâmide para tratar questões de multiescalas. Porém, um problema a ser solucionado é que o FAST não suporta variações na rotação.

Para superar essa deficiência o ORB utiliza uma medida, denominada de centróide de intensidade (*intensity centroid*) de uma região. Calcula-se o centróide de intensidade de uma região com um *keypoint* no centro. O vetor formado pelo *keypoint* e o centróide, dá uma medida de orientação θ. Para melhorar a invariância na rotação, momentos são calculados com as coordenadas x e y permanecendo dentro de uma área circular de raio r, onde r é o tamanho da região em questão.

Para computar os descritores dos pontos de interesses, o ORB utiliza o descritor BRIEF. Porém, o BRIEF não produz bons resultados com a ocorrência de rotações. Assim, o ORB direciona o BRIEF de acordo com a orientação dos pontos de interesses. Para cada conjunto de n pontos de interesse na localização (x_i, y_i), define-se uma matriz S de tamanho $2 \times n$, a qual contém as coordenadas desses pixels. Usando a orientação calculada para o FAST, uma matriz de rotação é encontrada e usada para rotacionar S. Assim, tem-se uma versão rotacionada S_θ de S.

O ORB discretiza o ângulo de rotação com incrementos de $2\pi/30$, e constrói uma tabela de pesquisa de padrões pré-computados do BRIEF. No momento em que a orientação do ponto de interesse θ for consistente, o conjunto de pontos S_θ será usado para computar seu descritor.

III. Especificações dos Testes

Os testes foram realizados com um *smartphone* com versão 4.1.2 do sistema operacional Android. O dispositivo possui uma câmera de 5M pixels e um processador Dual Core, de 1GHz e 1GB de memória RAM. No *smartphone* foi instalada a biblioteca OpenCV, a qual possibilitou a implementação rápida do sistema de software para a captura e processamento das imagens analisadas e para a avaliação dos algoritmos.

O experimento consistiu de uma seção de captura de um conjunto de imagens de uma determinada cena, com variações na rotação e escala. Após a etapa de captura das imagens, foi realizada uma etapa de *matching*, na qual é analisada a quantidade de descritores podem ser reconhecidos entre uma imagem e outra, considerando as variações citadas. Para isso, foram utilizados os algoritmos BFM (*Brute-Force Matcher*) e FLANN (*Fast Library for Approximate Nearest Neighbors*)[16]. Neste ponto, foram contabilizados os pontos com correspondências corretas, os pontos com correspondências erradas e o tempo necessário para a computação dos descritores e realização do *matching*.

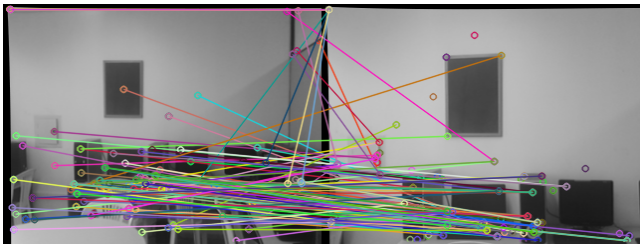
IV. Experimentos

Antes da captura das imagens foi feito um trabalho de calibração de câmera, a fim de que as possíveis distorções radiais causadas por imperfeições na lente da câmera pudessem ser corrigidas. Após esse tratamento prévio, foi capturado um conjunto de seis imagens de uma cena de um ambiente real. Durante a captura dessas imagens, variamos tanto a rotação da captura quanto na escala das imagens. Algumas imagens da cena capturada podem ser visualizadas na Figura **??**. Pode-se observar que, além das variações na rotação e na escala da imagens, existe também uma variação na iluminação. Assim, os algoritmos testados devem também mostrar robustez à variações na iluminação da cena.

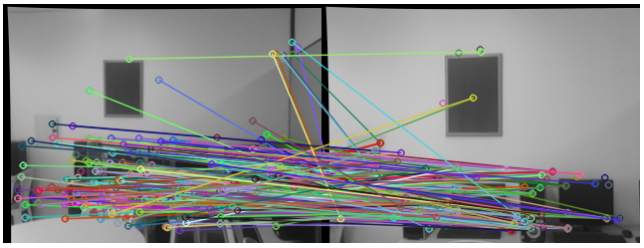
Figura 5. Exemplos de imagens capturadas da cena tratada.

Como especificado anteriormente, o próximo passo foi realizar o cálculo dos descritores com os algoritmos estudados e em seguida realizar o *maching*. Para isso, os descritores calculados para primeira imagem capturada foram comparados com os descritores de cada uma das imagens subsequentes do conjunto. As Figuras **??** e **??** mostram alguns dos teste do *matching* com os algoritmos BFM e FLANN, com os descritores calculados pelo SIFT, SURF e ORB. As linhas conectando pontos entres as imagens representam as correspondências entre descritores encontradas pelos algoritmos BFM e FLANN.

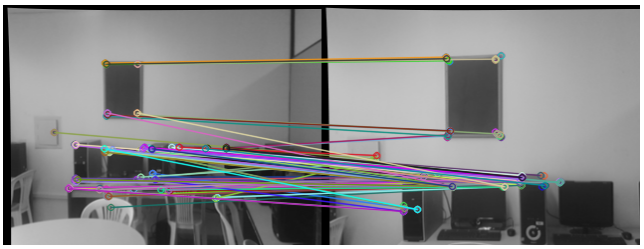
Os gráficos das Figuras 8 e 9 mostram os resultados obtidos nos testes. A Figura 8 a quantidade média de



(a) Matching de descritores SIFT.

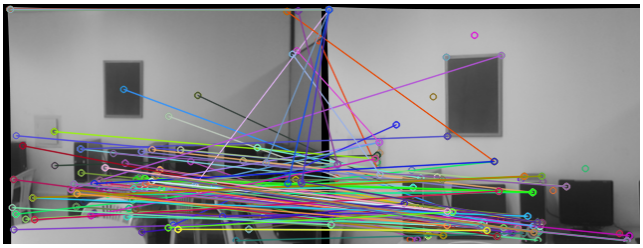


(b) Matching de descritores SURF.

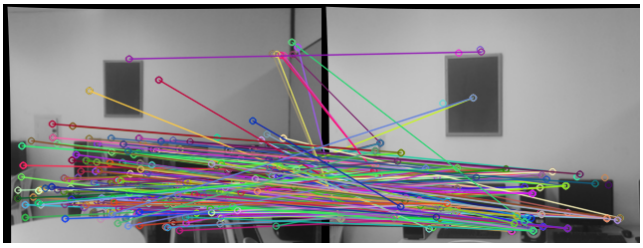


(c) Matching de descritores ORB.

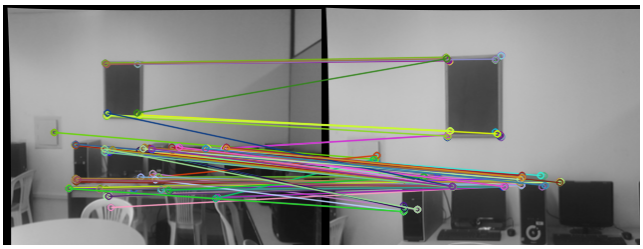
Figura 6. Correspondências estimadas pelo algoritmo BFM.



(a) Matching de descritores SIFT.



(b) Matching de descritores SURF.



(c) Matching de descritores ORB.

Figura 7. Correspondências estimadas pelo algoritmo FLANN.

descritores com boas correspondências e falsas correspondências, considerando que as correspondências foram analisadas para as 12 imagens do conjunto. A Figura 9 mostra o tempo necessário para computar os descritores e suas correspondências.

Esses resultados mostram que o algoritmos ORB se apresentou como o mais adequado para ser implementado em um sistema de visão computacional com detecção e correspondência de pontos notáveis com execução em tempo real. Assim, esses experimentos prévios, nos direciona a selecionar o OBR como o algoritmo a ser utilizado no sistema de navegação visual de um CellBot.

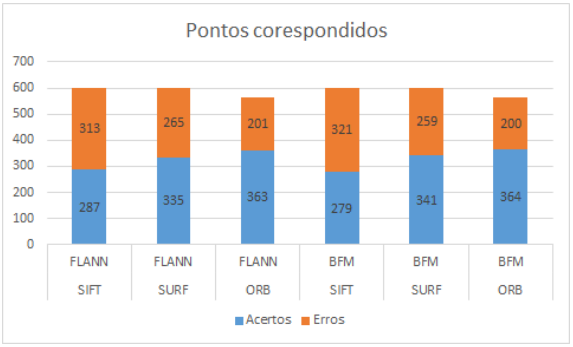


Figura 8. Quantidade de descritores calculados e suas correspondências acertadas e erradas.

V. Conclusão

Este trabalho apresentou um estudo sobre três dos principais algoritmos de detecção de pontos notáveis, para a implementação de um sistema de navegação visual para um robô baseado em *smartphone*. Os algoritmos estudados (o SURF, o SIFT e o ORB) foram testados em um *smartphone* com o sistema operacional Android munido da biblioteca de visão computacional OpenCV. Foram feitas análises de desempenho desses algoritmos com várias amostras de imagens mostrando a qualidade desses algoritmos na geração de seus descritores, que identificam pontos notáveis de uma determinada cena. Os testes mostraram que o algoritmo ORB apresentou os melhores resultados considerando a qualidade dos descritores e o tempo para o processamento das imagens.

Como trabalho futuro, serão realizados mais teste com cenas mais complexas, para nos direcionar a um resultado mais conclusivo. Após, o será implementado o sistema de navegação visual com um módulo de odometria visual e um módulo de Visual SLAM.

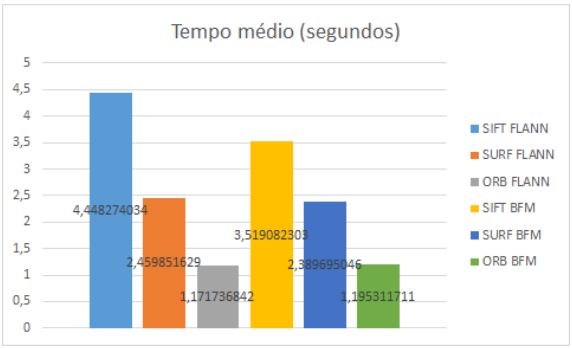


Figura 9. Tempo de processamento para cálculo de descritores e matching.

Referências

[1] A. A. S. Souza, "Mapeamento robótico 2,5-d com representação em grade de ocupação-elevação," Ph.D. dissertation, Universidade Federal do Rio Grande do Norte, 2012.

[2] R. Murphy, *Introduction to AI Robotics*, 1st ed. MIT Press, 2000.

[3] L. Greenwald and J. Kopena, "Mobile robot labs," *IEEE Robotics and Automation Magazine*, 2003.

[4] S. Piperidis, L. Doitsidis, C. Anastasopoulos, and N. Tsourveloudis, "Alow cost modular robot vehicle design for research and education," in *IEEE Mediterranean Conference on Control Automation (MED)*, Athens, Greece, 2007.

[5] H. Dinh and T. Inanc, "Low cost mobile robotics experiment with camera and sonar sensors," in *American Control Conference (ACC)*, St. Louis, USA, 2009, pp. 3793–3798.

[6] R. V. Aroca, R. B. Gomes, D. M. Tavares, A. A. S. Souza, A. M. F. Burlamaqui, G. A. P. Caurin, and L. M. G. Gonçalves, "Increasing students? interest with low cost cellbots," *IEEE Transaction on Education*, vol. 56, no. 1, pp. 3–8, 2013.

[7] A. C. Santos, L. Tarrataca, and J. M. P. Cardoso, "The feasibility of navigation algorithms on smartphones using j2me," *Mobile Networks and Applications*, vol. 15, no. 6, pp. 819–830, 2010.

[8] E. Guizzo and T. Deyle, "Robotics trends for 2012 [the future is robots]," *IEEE Robotics and Automation Magazine*, vol. 19, no. 1, pp. 119–123, 2012.

[9] L. Jian and L. Xiao-min, "Vision-based navigation and obstacle detection for uav," in *International Conference on Electronics, Communications and Control (ICECC)*, 2011.

[10] D. Scaramuzza, M. C. Achtelik, L. Doitsidis, F. Friedrich, E. Kosmatopoulos, A. Martinelli, M. W. Achtelik, M. Chli, S. Chatzichristofis, L. Kneip, D. Gurdan, L. Heng, G. H. Lee, S. Lynen, M. Pollefeys, A. Renzaglia, R. Siegwart, J. C. Stumpf, P. Tanskanen, C. Troiani, S. Weiss, and L. Meier, "Vision-controlled micro flying robots: From system design to autonomous navigation and mapping in gps-denied environments," *IEEE Robotics and Automation Magazine*, vol. 21, no. 3, pp. 26–40, 2014.

[11] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.

[12] J. Klippenstein and H. Zhang, "Quantitative evaluation of feature extractors for visual slam," in *Fourth Canadian Conference on Computer and Robot Vision*, Montreal, 2007, pp. 157–164.

[13] D. G. Lowe, "Object recognition from local scale-invariant features," in *IEEE International Conference on Computer Vision*, Kerkyra, 1999, pp. 1550–1157.

[14] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[15] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *IEEE International Conference on Computer Vision (ICCV)*, Spain, 2011.

[16] M. Muja and D. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *International Conference on Computer Vision Theory and Application*, 2009, pp. 331–340.