Análise de Desempenho de Detectores e Descritores de Características Utilizando a Plataforma Computacional Raspberry Pi

Samuel Rebouças de Jesus[1], **Walber Conceição de Jesus Rocha**[1]
João Carlos Nunes Bittencourt[1]

[1]Centro de Ciências Exatas e Tecnológicas
Universidade Federal do Recôncavo da Bahia
Caixa Postal 44380-000 – Bahia – BA – Brazil

{samuelreboucas07,walber_jesus}@hotmail.com, joaocarlos@ufrb.edu.br

Abstract. *The present work evaluates and compares the efficiency and computational performance of the feature detectors and feature description on the computational platform Raspberry Pi, algorithms possessing data referring to the precision, memory expense and runtime over the process of detection, description and correlation points of interest. The results presented become useful when designing systems with low computational resources, from the data produced it is possible to establish which detector algorithms and descriptors of features are proper according to systems constraints.*

Resumo. *Este trabalho consiste em avaliar a precisão e o desempenho dos algoritmos detectores e descritores de características na plataforma computacional Raspberry Pi, dispondo de dados referentes a precisão, consumo de memória e tempo de execução durante os processos de detecção, descrição e correlação dos pontos de interesse. Os resultados apresentados tornam-se úteis durante o processo de planejamento de sistemas aplicados em plataformas dotadas de baixo desempenho computacional, uma vez que, a partir dos dados produzidos é possível estabelecer quais algoritmos detectores e descritores de características são indicados baseado nas particularidades de um sistema.*

1. Introdução

Os seres humanos dispõem de recursos intrínsecos que os robôs não apresentam, dentre os quais destaca-se a visão. Ao longo das últimas décadas, técnicas e métodos de simulação da visão em sistemas robóticos têm sido desenvolvidos, permitindo que os mesmos possam extrair informações de modo a entender o ambiente que os cercam através do uso de câmeras e sensores. A visão computacional tornou-se uma tecnologia chave em vários campos, e tem sido utilizada como parte central de um vasto número de aplicações atualmente. Mais especificamente, a extração de características de imagens retrata uma área essencial da visão computacional. Dentro desse contexto, destacam-se os mecanismos de detecção de características (*feature detection*) e descrição de características (*feature description*). Tais abordagens apresentam algoritmos capazes de extrair características de imagens, transformando dados visuais em informações as quais sistemas computacionais são aptos a manipular [Awad and Hassaballah 2016].

Diversos métodos para detecção, descrição e correspondência de características em imagens foram propostos nos últimos anos [Ahmadabadian et al. 2013]. Uma diversidade de aplicações que utilizam desde computadores móveis até sistemas robóticos demandam por mecanismos de detecção de objetos [Schmidt et al. 2010a]. No entanto, em muitas aplicações as plataformas utilizadas não apresentam potencial computacional satisfatório. Por exemplo, aplicações robóticas que adotam plataformas embarcadas para a realização do processamento de imagens em campo [Fernanda 2018]. Neste sentido, torna-se necessária a realização de análises sistemáticas que produzam dados que expressem o desempenho computacional demandado por algoritmos detectores e descritores de imagens [Schmidt et al. 2010a].

O presente trabalho tem como objetivo apresentar dados que possibilitem realizar uma análise tanto do custo computacional produzido pelos algoritmos detectores e descritores de características, quanto do nível de precisão alcançado pelos mesmos. Para tanto foi utilizado a placa Raspberry Pi modelo B+, a qual tem sua utilização fundamentada ao baixo custo e a facilitade de vinculaçao entre a placa e a câmera. Portanto, ao relacionar os dados obtidos é possível mensurar quais as melhores opções de uso dentre os algoritmos escolhidos. Em vista disso foram realizados testes referentes a precisão, quantidade de memória alocada e tempo de execução dos algoritmos.

2. Detectores de Características

A utilização de detectores de características tem como objetivo identificar um conjunto de pontos de interesse a partir do processo de varredura e identificação de pixels de interesse em uma determinada imagem [Awad and Hassaballah 2016].

2.1. Speeded Up Robust Features

O método de detecção de características Speeded Up Robust Features (SURF), proposto por [Bay et al. 2006], apresenta um algoritmo rápido e robusto para extração e descrição de pontos de interesse de uma imagem. Para realizar a detecção dos pontos de interesse o SURF utiliza uma aproximação simples da matriz Hessiana, fazendo uso também de imagens integrais. O SURF utiliza uma aproximação com base em filtros de convolução. As derivadas Gaussianas de segunda ordem são aproximadas, assim sendo avaliadas rapidamente usando imagens integrais, independentemente do seu tamanho [Bay et al. 2006]. Devido ao uso de filtros de convolução e imagens em sua configuração integral, o SURF utiliza filtros de convolução com tamanhos diferentes diretamente na imagem original, produzindo assim o espaço de escalas, e consequentemente a invariância a mesma [Novais 2016].

2.2. Harris Corner Detector

O Harris Corner Detector (aqui denominado de Harris) é um detector de ponto de interesse largamente utilizado devido à sua forte invariância a rotação, escala, condições de iluminação e ruído de imagem. O Harris é baseado na função de autocorrelação local de um sinal. A medida de correlação é obtida relacionando uma região de pixel com seus fragmentos vizinhos, deslocadas por uma pequena quantidade de pixels em diferentes direções. Na presença de um ponto de interesse, a autocorrelação é alta para todas as direções de deslocamento [Schmid et al. 2000].

O algoritmo Harris utiliza como base a expansão de Taylor, a qual considera como entrada uma imagem em escala de cinza e a coordenada de um ponto, retornando um valor associado. Se o valor de retorno da expansão for um número negativo, o mesmo corresponde a um ponto de borda. Por outro lado, se o valor for positivo existe um ponto de superfície plana ou canto, dependente da magnitude deste valor [Bombardelli 2014].

2.3. Shi-Tomasi

O detector Shi-Tomasi é baseado no modelo de detecção proposto pelo método Harris. A diferença entre eles está na medida de canto, que passa a ser determinada a partir do menor dos valores absolutos dos autovalores do tensor estrutural [Schmidt et al. 2010b]. No método Shi-Tomasi, os cantos são os máximos locais da medida de canto. Essa medida é considerada mais robusta quando comparada com o método proposto pelo algoritmo Harris.

2.4. Features from Accelerated Segment Test

O método Features from Accelerated Segment Test (FAST) foi proposto com o objetivo de identificar pontos de interesse em uma imagem [Rosten and Drummond 2006]. O FAST é definido de modo a selecionar um pixel da imagem, o qual deve ser rotulado como um ponto de interesse ou não. Um valor de intensidade é definido e um círculo de 16 pixels é construído ao redor do pixel selecionado. Com isso, uma quantidade N de pixels contíguos dos 16 precisam estar acima ou abaixo da soma entre a intensidade do pixel e a intensidade limite determinada, para que o pixel selecionado possa então ser detectado como um ponto de interesse. De acordo com [Viswanathan 2009] o FAST não apresenta bons resultados para uma quantidade pequena de pixels contíguos, uma vez que podem ser detectados uma quantidade excessiva de pontos de interesse, reduzindo a sua precisão.

2.5. Oriented FAST and Rotated BRIEF

O Oriented FAST and Rotated BRIEF (ORB) é um algoritmo que surgiu como alternativa ao SURF, com o intuito de melhorar a sua eficiência em relação ao desempenho computacional [Rublee et al. 2011a]. O ORB é uma fusão do detector FAST (Features from Accelerated Segment Test) com o descritor Binary Robust Independent Elementary Features (BRIEF) [Santana et al. 2015].

Ao utilizar o ORB, primeiramente os pontos de interesse são detectados com base no FAST. Deste modo, uma medida de canto de Harris é aplicada para encontrar os N pontos principais [Karami et al. 2017]. Devido ao FAST não calcular a orientação, o detector ORB então calcula o centroide ponderado da intensidade do trecho com o ponto de interesse localizado no centro. Para melhorar a invariância de rotação, os momentos são calculados utilizando coordenadas x e y, as quais devem estar em uma região circular de raio r definida empiricamente como sendo o tamanho do trecho, de modo que x e y variem entre $[-r, r]$ [Rublee et al. 2011b, Santana et al. 2015].

2.6. Star

O algoritmo Star é um detector de ponto chave derivado do detector de características Center Surround Extrema (CenSurE) [Agrawal et al. 2008]. A principal motivação em volta do desenvolvimento deste detector está na obtenção da resolução espacial completa

em um detector multi-escala [Schmidt et al. 2010a]. O Star é baseado em filtros invariantes à rotação, onde o formato da máscara utilizada é formada por dois quadrados, um dos quais é rotacionado em 45 graus. A resposta do filtro, por sua vez, é calculada para sete escalas e para cada pixel da imagem. O tamanho da amostra é constante em cada escala e conduz a uma resolução espacial completa em todas as escalas. A magnitude da resposta do filtro fornece uma indicação da intensidade do recurso. Quanto maior o valor referente ao recurso, maior a probabilidade de ser repetitivo [Agrawal et al. 2008].

3. Descritores de Características

Um descritor de características é representado na forma de um conjunto de números, os quais, quando comparados, podem ser empregados no reconhecimento de objetos. As informações de uma imagem são utilizadas para caracterizar a aparência e formatos de grupos de pixels.

3.1. Scale-Invariant Feature Transform

O descritor Scale-Invariant Feature Transform (SIFT) extrai um conjunto de descritores de uma imagem, invariantes à translação, rotação e dimensão. O processo de descrição de pontos de interesse utilizando o algoritmo SIFT consiste de duas etapas principais: (i) atribuição de orientação; e (ii) descrição do ponto de interesse [Kabbai et al. 2015].

Durante a atribuição da orientação, cada ponto de interesse identificado na etapa de detecção é caracterizado com base em uma ou mais orientações. Essa orientação de ponto de interesse fornece um histograma de gradiente composto por 36 intervalos. Na etapa de descrição do ponto de interesse, o algoritmo propõe a criação de um conjunto de histogramas sobre uma janela de 16×16 pixels em torno do ponto de interesse. Descritores de pontos de interesse típicos utilizam 16 histogramas de orientação alinhados em uma grade 4×4. Cada histograma apresenta 8 variações de orientação, resultando em um vetor descritor com 128 elementos para cada ponto de interesse [Kabbai et al. 2013].

3.2. Binary Robust Independent Elementary Features

O Binary Robust Independent Elementary Features (BRIEF) é um descritor de características que usa testes binários simples entre pixels em um trecho de imagem suavizada. Entretanto, tal algoritmo apresenta alta sensibilidade à rotações no plano [Xu et al. 2012]. O BRIEF produz uma descrição da cadeia de bit, que consiste de uma série de valores representados pelos dígitos 0 e 1.

3.2.1. Oriented FAST and Rotated BRIEF

Para gerar seus descritores o algoritmo ORB utiliza o método BRIEF, avaliando as semelhanças entre os descritores utilizando a distância de Hamming. O resultado desta avaliação é refletido na forma de uma estrutura computacional rápida e flexível.O ORB modifica a estrutura do BRIEF de forma pontual, com o intuito de obter melhores resultados de detecção mesmo quando as imagens sofrem variações de rotação ou escala.

De acordo com sua definição, o ORB direciona o BRIEF de acordo com a orientação dos pontos de interesses [Santana et al. 2015]. 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 pelo detector, uma matriz de rotação é encontrada e usada para rotacionar S. Dessa forma, é possível obter uma versão rotacionada (S_θ) de S. Com base nisso, o ORB discretiza o ângulo de rotação com incrementos de $2\pi/30$, e constrói uma tabela de busca de padrões pré-computados a partir do BRIEF. No momento em que a orientação do ponto de interesse θ for consistente, o conjunto de pontos $S\theta$ será usado para computar seu descritor.

3.2.2. Binary Robust Invariant Scalable Keypoints

O descritor Binary Robust Invariant Scalable Keypoints (BRISK) foi proposto com o objetivo de alcançar um nível de qualidade comparável aos resultados obtidos a partir do método SURF, otimizando o seu tempo de processamento [Leutenegger et al. 2011]. Com o intuito de atingir melhores resultados de desempenho, o BRISK utiliza um padrão de amostragem manual, composto por anéis concêntricos, com mais pontos nos anéis externos. Dado um conjunto de pontos de interesse, o descritor BRISK implementa o método que identifica a direção característica de cada ponto. Isso o torna capaz de produzir descritores de orientação normalizada garantindo assim invariância à rotação [Leutenegger et al. 2011].

3.2.3. Fast Retina Keypoint

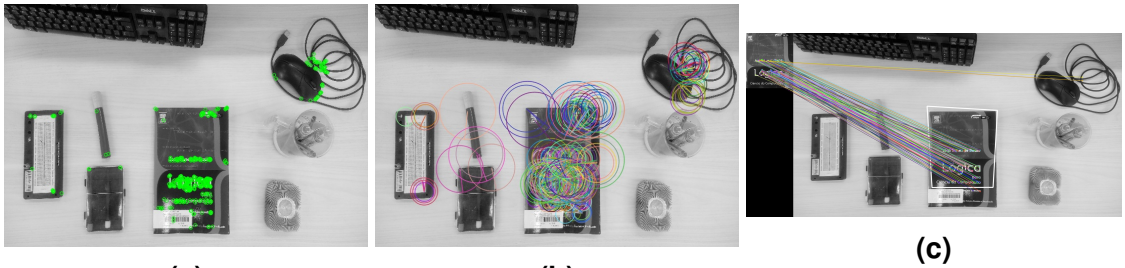
O Fast Retina Keypoint (FREAK) é um descritor de ponto chave inspirado na computação retiniana. Trata-se de um algoritmo compacto e robusto que utiliza um descritor binário construído a partir de um padrão de amostragem semelhante à configuração do sistema visual humano. No método FREAK, um conjunto de cadeias binárias é calculado a partir da comparação de pares de intensidades de imagem ao longo de um padrão de amostragem retinal [Vandergheynst et al. 2012]. Devido à robustez que o algoritmo empregado na implementação do método FREAK proporciona no que se refere a invariância à rotação, a orientação de trecho também pode ser estimada [Schmidt et al. 2013].

4. Ambiente de Avaliação dos Sistemas de Extração de Características

O ambiente de prototipação proposto neste trabalho envolveu a utilização da placa Raspberry Pi, modelo B+. O sistema de software foi desenvolvido em linguagem Python, versão 2.7, utilizado em conjunto com a biblioteca OpenCV, versão 3.2.0, para implementação das funções referentes à visão computacional e processamento de imagem. Foram escolhidos seis algoritmos detectores e cinco algoritmos descritores, resultando assim em trinta combinações de algoritmos de detecção/descrição de características.

O sistema de extração de característica foi dividido em três componentes principais. O primeiro consiste na utilização do algoritmo detector para obter os pontos de interesse na imagem analisada. A Figura 1a expressa uma imagem após a aplicação de um algoritmo detector. Cada círculo verde representa um ponto de interesse detectado. Posteriormente, é aplicado um algoritmo de descrição de características, representado na Figura 1b, onde cada ponto de interesse é representado por um círculo informando seu tamanho e orientação.

Uma vez que o presente trabalho visa fornecer informações de desempenho para aplicações de detecção e rastreio de objetos, torna-se necessário realizar a comparação de imagens, com o propósito de obter pontos que possuam as mesmas características. Para tanto, optou-se pelo uso do método força bruta, denominado BFMatcher [OpenCV 2014]. Este método utiliza os dados referentes ao descritor de uma imagem de referência, relacionando-o com o descritor da imagem a ser analisada.



(a) (b) (c)

Figura 1. Caracterização e correlação de uma imagem.

A Figura 1c expressa a utilização do correspondente de força bruta BFMatcher entre uma imagem de referência (à esquerda) e uma imagem a ser identificada. Mediante a utilização dos dados retornados do descritor, torna-se possível obter os pontos correspondentes entre as imagens. Com base nesses dados, é possível obter uma métrica de precisão para um dado algoritmo. Para cada imagem utilizada, a região a qual o objeto se encontra é diferente. Portanto, as regiões foram previamente definidas e concedidas como parâmetro de entrada ao sistema de análise. Quando uma correspondência é determinada como satisfatória a mesma é computada como correta. Ao final deste processo, a quantidade de correspondências obtidas e a quantidade de correspondências corretas alcançadas para cada uma das treze imagens presentes no *dataset* são definidas. Após a realização desta análise, um conjunto de testes de desempenho foi realizado, com o objetivo de obter um perfil de execução para cada combinação.

A realização dos testes de desempenho produziu dados referentes à quantidade de memória alocada e ao tempo de execução dos algoritmos, proporcionando assim uma análise quantitativa geral de execução de cada combinação. Estes testes permitiram ainda analisar cada algoritmo, definindo assim os pontos responsáveis por limitar o desempenho dos mesmos.

5. Resultados

Os testes realizados ao longo desta pesquisa obtiveram dados relacionados a quantidade de correspondências obtidas, a quantidade de correspondências corretas, quantidade de pontos de interesse detectados e porcentagem de rastreio entre as imagens testadas. Entretanto, analisar apenas estes dados isoladamente não possibilita realizar uma análise contundente a respeito da qualidade das correspondências. Portanto relacionou-se os dados de forma a produzir valores percentuais referentes a relação entre quantidade de correspondências obtidas e correspondências corretas, sendo estes dados expressos na Tabela 1.

	BRIEF	BRISK	FREAK	ORB	SIFT
FAST	**89,54**	63,54	76,72	**82,21**	94,05
Harris	76,83	48,18	63,01	73,64	41,18
ORB	**94,62**	93,04	99,00	**92,15**	93,18
Shi-Tomasi	75,05	69,20	69,35	69,15	68,89
Star	71,09	54,34	67,03	60,35	81,00
SURF	73,59	78,05	71,54	64,00	94,92

Tabela 1. Percentual de acertos dos pares de algoritmos.

Foram geradas treze imagens para cada par de algoritmo, de forma que cada imagem evidencie as correspondências obtidas. A Figura 2a representa o resultado de uma imagem analisada pelo detector ORB em conjunto com o descritor FREAK. Verifica-se que as correspondências obtidas estão concentradas no objeto alvo da análise. A partir desta análise, foi constatado que percentual de precisão referido na Tabela 1 é coerentes com a imagem resultante da execução de um dado par de algoritmos.

(a) **(b)** **(c)**

Figura 2. Amostra de resultados obtidos durante o teste de exatidão.

A Figura 2b se refere a junção dos algoritmos Star e BRIEF. Pode-se notar como as correspondências obtidas estão dispersas, uma vez que este par apresentou $71,09\%$ de acerto em relação às correspondências detectadas. No entanto, observa-se que, apesar da captura das falsas correspondências, o objeto foi selecionado corretamente, não afetando assim a detecção do mesmo. Por fim, a Figura 2c representa a combinação entre os algoritmos Harris e SIFT. Observa-se que as correspondências estão, em sua maioria, dispersas e equivocadas, corroborando com o percentual de $41,18\%$ de acerto indicado na Tabela 1. Devido a baixa qualidade dos dados obtidos, a identificação correta do objeto foi comprometida.

A Tabela 2 apresenta os dados referentes ao consumo médio de memória em função de cada combinação entre os detectores e descritores. Ao analisar a influência dos descritores observa-se um padrão na distribuição dos dados, mais especificamente os descritores BRIEF e ORB foram responsáveis por apresentar os menores valores de consumo de memória. Sendo assim, uma vez que o descritor ORB é baseado no BRIEF, com modificações responsáveis por garantir invariância a rotação e escala, o custo computacional produzido por ambos não sofreu variações significativas ao longo dos testes. Na Tabela 2, nota-se ainda a influência negativa do descritor BRISK, uma vez que, ao ser utilizado em conjunto com qualquer detector resulta nos maiores índices de consumo de memória. Durante os testes, verificou-se que este excedente relativo ao consumo de memória é decorrente da função de instanciação do descritor BRISK presente na biblioteca do OpenCV.

Após uma análise aprofundada, verificou-se que cerca de 90% do total de memória é alocada ao instanciar o BRISK.

	Memória Consumida (MB)				
	BRIEF	**BRISK**	**FREAK**	**ORB**	**SIFT**
FAST	**3,28**	48,20	11,27	**3,07**	7,48
Harris	11,78	56,75	19,75	11,50	12,29
ORB	**3,67**	48,58	11,63	**3,46**	10,42
Shi-Tomasi	3,81	48,84	11,85	3,53	6,75
Star	3,14	48,22	11,17	3,13	6,48
SURF	3,84	48,85	11,83	3,94	10,29

Tabela 2. Quantidade de memória consumida ao longo da execução do algoritmo.

A Tabela 3 expressa os valores obtidos referentes ao tempo médio gasto por cada junção do algoritmo detector e descritor.

Algoritmo	**Tempo de execução (s)**				
	BRIEF	**BRISK**	**FREAK**	**ORB**	**SIFT**
FAST	**1,31**	8,18	2,31	**0,86**	17,83
Harris	1,43	7,84	2,26	1,52	3,82
ORB	**0,95**	7,38	1,54	**1,46**	29,24
Shi-Tomasi	1,24	7,63	2,08	1,29	3,55
Star	3,60	9,96	4,42	3,64	11,87
SURF	6,91	13,56	7,22	6,98	109,41

Tabela 3. Tempo médio de execução dos algoritmos.

Observa-se, na Tabela 3, que os resultados referentes aos descritores BRIEF e ORB retornam os menores valores de tempo gasto para a maioria dos detectores utilizados. Isso ocorre devido ao fato de que estes descritores apresentam também os menores valores referentes ao consumo de memória. Em contrapartida, os algoritmos que utilizam os descritores BRISK e SIFT apresentaram os maiores valores de tempo de execução, independente do detector utilizado. O descritor BRISK apresentou elevado tempo de execução devido ao excessivo consumo de memória. Por exemplo, o descritor BRISK, ao ser utilizado em conjunto com o detector FAST, gastou, em média, $6,33$ segundos para instanciar o descritor (mesma função responsável por consumir maior quantidade de memória). Este tempo, por sua vez, representa cerca de $77,38\%$ do tempo médio de execução do referido algoritmo.

Ao analisar os melhores casos, em relação ao tempo de execução, verifica-se que o algoritmo proveniente do detector FAST e descritor ORB é $10,46\%$ mais rápido, quando comparado com a combinação que utilizou o detector ORB e o descritor BRIEF, e $44,19\%$ mais rápido que a combinação entre o detector Shi-Tomasi e descritor ORB. Por outro lado, ao relacionar os dados referentes ao consumo de memória, juntamente com o tempo gasto durante a execução de cada algoritmo, destacam-se positivamente os descritores BRIEF e ORB. Os algoritmos provenientes da junção entre um detector com algum desses dois descritores apresentaram as menores demandas de memória e os menores tempos de execução.

6. Conclusões

Nas aplicações de visão computacional, o desempenho e a precisão são fatores determinantes no sentido de expressar a qualidade de um dado sistema, apresentando-se assim como uma problemática ainda maior quando a aquisição dos resultados dependem do fator tempo. Em vista disso, ser capaz de analisar e extrair características de imagens a uma taxa capaz de atender aos requisidos de tempo real de uma aplicação embarcada, ao tempo em que propicie fluidez ao sistema, apresenta demasiada importância para sistemas embarcados de visão computacional.

A partir do desenvolvimento deste trabalho, foi possível realizar uma análise particular a respeito de algoritmos aos quais os dados produzidos se destacaram em relação aos demais. Constatou-se a influência que os detectores exercem sobre a quantidade de correspondências capturadas. Do mesmo modo, observou-se a influência que os descritores impõem sobre o tempo de execução dos algoritmos. A partir da análise dos dados de eficiência e desempenho foram definidos as combinações de algoritmos que obtiveram os melhores resultados relacionados à precisão, tempo de execução e consumo de memória. Neste sentido, as combinações entre detector/descritor FAST-ORB, ORB-BRIEF, FAST-BRIEF e ORB-ORB apresentaram resultados que expressam uma qualidade superior em relação às demais combinações. Entretanto, a principal limitação destas análises realizadas caracteriza-se pela ausência de variância em relação a rotação, luminosidade e distância nas imagens utilizadas. Deste modo, a análise de precisão e desempenho dos algoritmos é restrita a um cenário ideal. A partir do desenvolvimento deste trabalho, será possível construir uma plataforma de rastreio de objetos na qual propriedades como rotação, luminosidade, distância e ruído não influenciem no processo de correlação das imagens. Vislumbra-se ainda a elaboração de um sistema de rastreio genérico, baseado em técnicas de aprendizado de máquina.

Referências

Agrawal, M., Konolige, K., and Blas, M. R. (2008). Censure: Center surround extremas for realtime feature detection and matching. In *European Conference on Computer Vision*, pages 102–115. Springer.

Ahmadabadian, A. H., Robson, S., Boehm, J., Shortis, M., Wenzel, K., and Fritsch, D. (2013). A comparison of dense matching algorithms for scaled surface reconstruction using stereo camera rigs. *ISPRS Journal of Photogrammetry and Remote Sensing*, 78:157–167.

Awad, A. I. and Hassaballah, M. (2016). *Image feature detectors and descriptors: foundations and applications*, volume 630. Springer.

Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer.

Bombardelli, F. G. (2014). Localização de robôs móveis por aparência visual.

Fernanda, J. (2018). Sistema de navegação autônoma para plataforma robótica móvel com restrições não-holonômicas. B.S. thesis, Universidade Estadual de Londrina.

Kabbai, L., Abdellaoui, M., and Douik, A. (2013). Hybrid classifier using sift descriptor. In *Control, Decision and Information Technologies (CoDIT), 2013 International Conference on*, pages 388–392. IEEE.

Kabbai, L., Azaza, A., Abdellaoui, M., and Douik, A. (2015). Image matching based on lbp and sift descriptor. In *Systems, Signals & Devices (SSD), 2015 12th International Multi-Conference on*, pages 1–6. IEEE.

Karami, E., Prasad, S., and Shehata, M. (2017). Image matching using SIFT, SURF, BRIEF and ORB: performance comparison for distorted images. *arXiv preprint arXiv:1710.02726*.

Leutenegger, S., Chli, M., and Siegwart, R. Y. (2011). Brisk: Binary robust invariant scalable keypoints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2548–2555. IEEE.

Novais, J. P. (2016). Aplicação dos Algoritmos SIFT e SURF na Classificação de Sub-Imagens por Discriminação de Textura.

OpenCV (2014). Feature matching.

Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer.

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

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011b). ORB: An efficient alternative to SIFT or SURF. In *Computer Vision (ICCV), 2011 IEEE international conference on*, pages 2564–2571. IEEE.

Santana, B. A., Maia, R. d. S., Figuerola, W. B., and Souza, A. A. (2015). Análise de desempenho de algoritmos detectores de keypoints para um sistema de navegaçao visual de robôs baseados em smartphones.

Schmid, C., Mohr, R., and Bauckhage, C. (2000). Evaluation of interest point detectors. *International Journal of computer vision*, 37(2):151–172.

Schmidt, A., Kraft, M., Fularz, M., and Domagała, Z. (2013). Comparative assessment of point feature detectors in the context of robot navigation. *Journal of Automation Mobile Robotics and Intelligent Systems*, 7.

Schmidt, A., Kraft, M., and Kasiński, A. (2010a). An evaluation of image feature detectors and descriptors for robot navigation. In Bolc, L., Tadeusiewicz, R., Chmielewski, L. J., and Wojciechowski, K., editors, *Computer Vision and Graphics*, pages 251–259, Berlin, Heidelberg. Springer Berlin Heidelberg.

Schmidt, A., Kraft, M., and Kasiński, A. (2010b). An evaluation of image feature detectors and descriptors for robot navigation. In *International Conference on Computer Vision and Graphics*, pages 251–259. Springer.

Vandergheynst, P., Ortiz, R., and Alahi, A. (2012). Freak: Fast retina keypoint. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 510–517. Ieee.

Viswanathan, D. G. (2009). Features from accelerated segment test (fast).

Xu, J., Chang, H.-w., Yang, S., and Wang, M. (2012). Fast feature-based video stabilization without accumulative global motion estimation. *IEEE Transactions on Consumer Electronics*, 58(3).