Universidade Federal
de Campina Grande

Centro de Engenharia Elétrica e Informática
Curso de Graduação em Engenharia Elétrica

Testes de Desempenho de Algorítimos de Deteção de Artefatos em Plataforma Embarcada Raspberry Pi Utilizando OpenCV e Node.JS

Arthur de Araújo Farias

Campina Grande, PB - Brasil

23 de março de 2018

ARTHUR DE ARAÚJO FARIAS

Testes de Desempenho de Algorítimos de Deteção de Artefatos em Plataforma Embarcada Raspberry Pi Utilizando OpenCV e Node.JS

Trabalho de conclusão de curso para graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, em exigência da disciplina de Projeto de Conclusão de Curso.

Orientador: Prof. Dr. Alexandre Cunha de Oliveira

Campina Grande, PB - Brasil

23 de março de 2018

Arthur de Araújo Farias

Testes de Desempenho de Algorítimos de Deteção de Artefatos em Plataforma Embarcada Raspberry Pi Utilizando OpenCV e Node.JS/ Arthur de Araújo Farias. – Campina Grande, PB - Brasil, 23 de março de 2018-

47 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Alexandre Cunha de Oliveira

Trabalho de Conclusão de Curso – Centro de Engenharia Elétrica e Informática
Curso de Graduação em Engenharia Elétrica, 23 de março de 2018.

1. Visão computacional. 2. Sistemas Embarcados. 2. *feature matching*. I. Alexandre Cunha Oliveira. II. Universidade Federal de Campina Grande. III. Departamento de Engenharia Elétrica. IV. Teste de Desempenho de Algoritimos de Detecção de Artefatos em Plataforma Embarcada Raspberry Pi Utilizando OpenCV e Node.JS

Arthur de Araújo Farias

Testes de Desempenho de Algorítimos de Deteção de Artefatos em Plataforma Embarcada Raspberry Pi Utilizando OpenCV e Node.JS

Trabalho de conclusão de curso para graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, em exigência da disciplina de Projeto de Conclusão de Curso.

Trabalho aprovado. Campina Grande, PB - Brasil, 24 de novembro de 2012:

Prof. Dr. Alexandre Cunha de Oliveira
Orientador

Prof. Dr. Marcos R. A. Morais
Avaliador

Campina Grande, PB - Brasil
23 de março de 2018

Este trabalho é dedicado a todos aqueles que se empenharam e paciência tiveram para ver-me um engenheiro.

Agradecimentos

Os agradecimentos principais são direcionados a Paulo de Almeida Farias e Antônia de Araújo Farias pela paciência infinita e a boa vontade. Ao meu amigo e chefe Igor Silva pela paciência, disponibilidade e esforços em fazer tudo isso acontecer. Aos meus professores Alexandre Cunha e Marcos Morais pela cobrança pela perfeição e um trabalho digno de excelência. Agradecimentos especiais a Universidade Federal de Campina Grande por oferecer sempre o melhor em termos de educação e formação.

Agradeço também a todos os que abriram as portas para que pudesse eu demostrar todo o meu potencial com muito empenho e dignidade.

"What we know is a drop, what we don't know is an ocean. (Isaac Newton)

Resumo

Este trabalho de conclusão de curso contempla o desenvolvimento de um sistema de experimentação de três métodos de *feature matching*: ORB, SIFT e SURF utilizados como base para técnicas de reconhecimento de objetos utilizados por sistemas de visão computacional. O sistema consiste em um software embarcado em uma plataforma Raspberry Pi, microprocessada por um núcleo ARM Cortex A53. O sistema utiliza um sistema operacional Linux e foi submetido a experimentação no que tange ao tempo de processamento.

Foi constatado experimentalmente que o tempo de processamento pela técnica ORB é muito mais rápio que o método SIFT. Sendo que o desempenho com respeito a taxa de acertos em relação a figura é semelhante ao do método SIFT e SURF. O método SURF possui um desempenho, em termos de processamento ligeiramente inferior ao método ORB, mas com a grande desvantagem de ser um algoritimo protegido por patentes.

Palavras-chaves: Visão computacional, sistemas embarcados, *feature matching*, ORB, SIFT, SURF, Raspberry Pi.

Abstract

This work of completion of course contemplates the development of a system of experimentation of three methods of feature matching: ORB, SIFT and SURF used as a basis for object recognition techniques used by computer vision systems. The system consists of software embedded in a Raspberry Pi platform, microprocessed by an ARM Cortex A5O que são3 core. O system uses a Linux operating system and has undergone experimentation in the processing time.

It has been found experimentally that the processing time by the ORB technique is much faster than the SIFT method. Being that the performance with respect to the rate of correctness in relation to the figure is similar to that of the SIFT SURF. The SUFT method performs in terms of processing slightly lower than the ORB method, but with the great disadvantage of being a algorithm protected by patents.

Keywords: Computer Vision, Embedded Systems, *feature matching*, ORB, SIFT, SURF, Raspberry Pi.

Lista de ilustrações

Lista de abreviaturas e siglas

ORB	*Oriented FAST and Rotated BRIEF*
SIFT	*Scale Invariant Feature Transform*
SURF	*Speeded-Up Robust Features*
OpenCV	*Open Source Computer Vision Library*
INRIA	*French Institute for Research in Computer Science and Automation*
SUSAN	*Standing for smallest univalue segment assimilating nucleus*
FAST	*Features from Accelerated Segment Test*
LoG	*Laplacian of Gaussian*
HDMI	*High-Definition Multimedia Interface*
USB	*Universal Serial Bus*
SD	*Secure Digital*
SoC	*System on Chip*
ARM	*Advanced RISC Machine*
GPIO	*General Purpose Input-*

Sumário

1 Introdução

Neste capitulo introduz-se os conceitos básicos os quais este trabalho fundamenta-se. Primeiro, visita-se o conceito de visão computacional e o uso desta ciência no mundo real. Subsequentemente, explica-se o campo do processamento digital de imagens, disciplina que relaciona-se intimamente com o tema visão computacional e é base para as técnicas de *feature detection* e *feature matching*, alvo da comparação de desempenho veículada por este trabalho.

1.1 Visão Computacional

A visão é um recursos que os seres vivos usam para o posicionamento e contextualização semântica durante o tempo em que permanecem vivos. A adição de significado aos elementos do fluxo contínuo de imagens é parte fundamental da existência. A visão computacional é a disciplina que tenta construir técnicas para, de forma objetiva processar e atribuir contexto a imagens obtidas por sistemas eletrônicos computacionais.

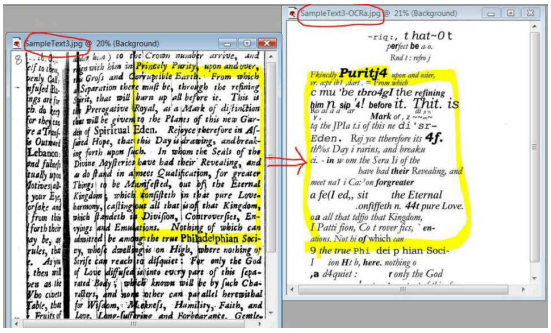
A disciplina de visão computacional é amplamente estudada por muitos centro de pesquisas e empresa em todo o mundo. Técnica mais variadas e diversos modelos matemáticos que servem aos mais diversos propósitos vêm sendo desenvolvidos recentemente. Atualmente dispõe-se de técnicas para reconstrução parcial de modelos tridimensionais através do processamento de imagens bidimensionais, obteve-se sucesso no parcial reconhecimento de objetos em uma cena bidimensional e muitas outras conquistas. Porém a tarefa de atribuição de contexto da forma que os seres humanos conseguem realizar é uma tarefa que configura-se um campo aberto nesta ciência.

Quais desafios o campo têm encontrado nos últimos anos? Em parte, por conta das limitações em processamento. Exsistem gargalos na capacidade dos sistemas em processarem a quantidade de informação necessária.

As mais proeminentes áreas de investigação em visão computacional nos dias de hoje são:

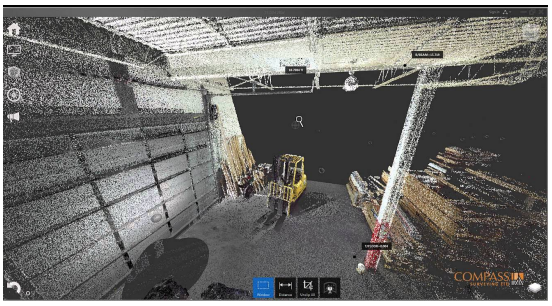
Reconhecimento óptico de Caracteres (OCR): transformação de imagens de texto em texto, como o reconhecimento de placas de automóveis, livros e códigos postais escritos a mão fazem parte deste campo.

Inspeção de Máquinas: a rápida inspeção de máquinas para determinação de qualidade da produção usando visão estéreo com iluminação especializada tem sido utilizada na industria para substituir a inspeção visual de produtos.



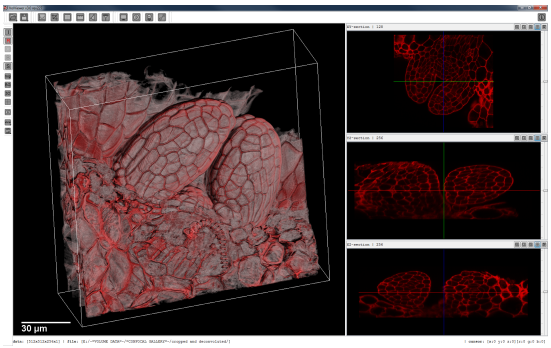
(a) Reconhecimento de Caracteres

Fonte: <www.passtheword.org>



(b) Inspeção de Máquinas

Fonte: <youtube.com>



(c) Imagens Médicas

Fonte: <http://biii.eu>

(d) Captura de movimentos

Fonte: <youtube.com>

Figura 1 – Exemplos de aplicações de sistemas de visão computacional.

Imagens Médicas: registrar imagens em procedimentos médicos, para realização de estudos a longo prazo têm sido uma grande e vasta área de aplicação da visão computacional para os mais diversos métodos de aquisição de imagens.

Segurança Automotiva: a previsão de colisões por sistemas de processamento em tempo real de imagens têm sido utilizado por automóveis em sistemas de detecção e mitigação de colisões.

Captura de Movimentos: as mais diversas industrias buscam a captura de movimento para tentar construir modelos que sirvam para melhorar a desempenho de seus produtos. A industria cinematográfica, por exemplo, utiliza a captura de movimentos para construir produtos audiovisuais cada vez mais imersivos.

Muitas outras áreas da atividade humana tem usado os sistemas de visão computacional, as aplicações estão expandindo e melhorando cada vez mais diversas áreas dos produtos tecnológicos humanos. A imersão destas tecnologias nas atividades cotidianas serão cada vez mais comuns de forma a mudar cada vez mais as formas tem-se produzido tecnologia e cultura.

1.2 Processamento Digital de Imagens

Uma imagem pode ser definida como a função bidimensional, $f(x, y)$, onde x e y é um plano espacial de coordenadas e a amplitude f em qualquer par de coordenadas (x, y) é denominado intensidade ou nível de escala de cinza no ponto (x, y). Quando x, y e o valor de intensidade f são todos finitos e enumeráveis, denominamos a imagem de imagem digital. O campo do processamento digital de imagens refere-se ao processamento de imagens digitais por sistemas computacionais. Deve-se notar que uma imagem digital é composta por um número finito de elementos, cada um possui uma localização e valor. Esses elementos são denominados *pixels*.

A visão é um dos sentidos mais presentes nas vidas humanas, não é de surpreender-se que é uma das principais partes da percepção humana. Porém, diferentemente dos humanos, que limita-se a banda visual eletromagnética entre 430-750 THz. Os sistemas de visão computacional dependem apenas da tecnologia eletrônica empregada. As possibilidades se expandem por quase todo o espectro eletromagnético.

Não há um acordo geral entre pesquisadores de quais são as fronteiras do processamento digital de imagens em relação a análise de imagens e visão computacional. Algumas vezes a distinção é feita definindo processamento de imagens como uma disciplina na qual as entradas e saídas do objeto de processamento a ser estudado são imagens. Observa-se o fato de que o escopo deste trabalho é a análise de desempenho das bases de uma técnica de reconhecimento de objetos em imagens. As entradas do sistema em si são imagens e a saída do sistema são posições em um plano bidimensional de *features* correlacionados a uma imagem de referência.

1.3 Detecção de Objetos

Detecção de objetos é o processo de encontrar instâncias de objetos no mundo real como faces, bicicletas, construções... Geralmente utiliza-se dados gráficos como vídeos e fotos. Os algoritmos de reconhecimentos geralmente utilizam extração de *features* ou aprendizado de máquina. As aplicações são as mais variadas, desde sistemas segurança a entretenimento.

Consiste a detecção de objetos em um dos problemas fundamentais nos sistemas de visão. Diversas técnicas foram desenvolvidas para resolver o referido problema. Essas técnicas incluem Detecção de Objetos baseada em correspondência de *features*, Detecção de Objetos pelo método Viola-Jones, Classificação SVM com gradientes orientados a histograma e detecção de blobs.

O método Viola-Jones foi um método proposto em 2001 por Paul Viola e Michael Jones para detecção de faces em tempo real para streams de vídeo e posteriormente foi am-

plamente estendido a outras finalidades(VIOLA; JONES, 2001). No problema da detecção de faces, um olho humano pode facilmente realizá-lo. Porém, um sistema computacional não possui os mesmos recursos que a mente humana dispõe. Deve-se, portanto, encarar o problema de uma forma alternativa a fim de garantir a detecção. O algoritmo em si possui diversas características interessantes como: alta taxa de acertos e tempo real.

A metodologia de classificação SVM com gradientes orientados a histogram baseada-se em descritores de *features*. A técnica consiste em contar ocorrências de orientações em porções localizadas de uma imagem. O método foi inicialmente descrito por Robert K. McConnell da Wayland Research Inc em 1986 sem a denominação aqui discutida(MCCONNELL, 1986). O conceito foi utilizado posteriormente pela Mitsubishi Electric Research Laboratories(FREEMAN; ROTH, 1994), porém, só foi amplamente difundido em 2005 quando Navneet Dalal e Bill Triggs, pesquisadores do Instituto Nacional Francês em Ciências da Computação e Automação (INRIA), apresentaram seu trabalho suplementar na Conferência em Visão Computacional e Reconhecimento de Padrões (CVPR). O trabalho focou-se na detecção de pedestres em imagens estáticas.

A detecção de blobs, por sua vez, consiste em detectar regiões em uma imagem com diferentes propriedades como brilho ou cor, comparada com regiões ao redor.Informalmente um blob é uma região de uma imagem em que as propriedades dessa região são constantes ou quase constantes. Todos os pontos em um blob podem ser considerados similares. O método mais comum de detecção de blobs é por convolução.

Recentemente, detecção de objetos com *features* locais em imagens emergiu como uma técnica efetiva no reconhecimento de objetos. Várias metodologias têm surgido com o propósito de resolver diferentes problemas. Para um olhar mais profundo sobre o tema, faz-se necessário a descrição de alguns conceitos: Detecção de ponto, Descrição de pontos e Correspondência de Pontos.

A detecção de pontos consiste em determinar a localização de pontos de interesse em uma imagem. A descrição de pontos corresponde em determinar expressões representativas dos pontos de interesse em uma imagem. A correspondência de pontos consiste em determinar similaridades ou dissimilaridades para os descritores de forma a encontrar pontos de interesses relacionados em duas imagens.

O método a ser escolhido para detecção de pontos de interesse depende basicamente da aplicação a ser utilizada. Faz-se necessário a realização de testes sobre as diversas metodologias disponíveis. Assim como a detecção, as metodologias de descrição e correspondência devem ser provados a fim de garantir uma boa estratégia para uma aplicação específica.

2 Fundamentação Teórica

Neste capitulo introduz-se os conceitos relacionados a *image features, descriptors, detectors*, que são técnicas utilizadas para extrair informações uteis de imagens, e os sistemas computacionais normalmente utilizados para processar algoritimos de analise de imagens.

Os algoritimos presentes para realizar o reconhecimento de objetos vão desde formas abstratas desenvolvidas levando em consideração algumas características morfológicas das próprias imagens, outras partem da forma em que os sistemas neurais humanos realizam a tarefa de classificação de imagens. O trabalho aborda três técnicas morfológicas com forte base analítica matemática para comparação entre imagens denominada *feature matching*.

2.1 *Image Features*

O conceito de *feature* é construído para denotar um conjunto de pixels relevantes na identificação única de um objeto real em uma imagem. Eles utilizam características elementares de imagens como a forma, a cor, a textura e movimento para construir uma identificação única de uma região e garantir que certas transformações decorrente das possibilidades de captura, como variação em escala e transformações morfológicas sejam irrelevantes. Constrói-se descritores dessas características para comparação com outras imagens.

A título de exemplo, o problema em questão é encontrar a Figura 2a na Figura 2b. As *features* de ambas as imagens, geram descritores que serão comparados a fim de estabelecer relações entre as duas imagens.

2.2 *Image Descriptors*

Alguns detectores de *features* estão prontos para utilizar informações de saída para que algoritimos possam estabelecer relações entre *features* de outra imagem. Este é o caso dos algoritimos SIFT e SURF. Existem detectores menos eficientes como LESH (Local Energy based Shape Histogram) ou GLOH (Gradient Location and Orientation Histogram).

Quando trabalha-se com detectores que geram descritores, o próximo passo consiste em encontrar correspondências. Em alguns casos, algoritimos simples como a soma das diferenças ao quadrado ou a correlação cruzada normalizada pode ser utilizada para

(a) Imagem do objeto em analise

Fonte: <www.mathworks.com>

(b) Imagem a localizar o objeto

Fonte: <www.mathworks.com>

Figura 2 – Problema básico referente às técnicas de *feature extraction* e *feature matching*.

realizar correspondências.

Na maioria dos casos, porém, a aparição de *features* mudará a orientação e escala. Muitas vezes, inclusive, transformações afim também faz parte dos fatores a serem considerados em um algoritimo.

2.3 Metodologias de Extração de *Features* em Imagens

A detecção de *features* em imagens incluem métodos computacionais de abstração de informações em imagens e realização de decisões locais em cada ponto de uma imagem em que há um artefato ou não. O resultado da operação de extração de *features* consiste em um subconjunto no domínio da imagem em forma de pontos, curvas ou regiões.

Formalmente não existe uma definição do que é um artefato em uma imagem, a definição de um artefato depende do tipo de problema a ser enfrentado ou aplicação a ser desenvolvida. Mesmo assim, tenta-se generalizar o conceito como uma região ponto ou curva a ser detectada a fim de extrair alguma informação abstrata a respeito de uma imagem. Como exemplo, um rosto pode ser uma entidade abstrata a ser detectada em uma imagem e as características a serem extraídas da imagem de forma que sirvam a identificação do rosto são denominados *features*.

A detecção de *features*, portanto consiste em uma operação de baixo nível que serve para diversos algorítimos para extração de informações abstratas. Esta, portanto é a primeira operação a ser realizada sobre o domínio de uma imagem com a finalidade de obtenção de informações.

Os tipos de *features* a serem extraídos em imagens consistem basicamente em bordas bordas, contornos ou regiões. As mais diversas técnicas são apresentadas neste trabalho a fim de introduzir a maioria das tecnologias disponíveis para realização de

extração de *features*.

2.3.1 *Edge Detection*

Detecção de contornos incluem uma variedade de métodos matemáticos que identificam pontos em uma imagem digital em que a mudança de brilho muda descontínuamente. O conjunto de pontos que formam uma curva onde o contraste muda rapidamente é denominado contorno. Aqui prefere-se a utilização do termo em inglês, *edge* por praticidade e conformidade com a maior parte da literatura.

Figura 3 – Exemplo de imagem submetida a um algoritimo de detecção de contornos.

Fonte: <https://en.wikipedia.org>

Deve-se de antemão determinar que o mesmo problema de encontrar descontinuidades em sinais unidimensionais é denominado detecção de degraus. Muitas das metodologias para detecção de contornos baseia-se em generalizações das metodologias unidimensionais.

Geralmente uma descontinuidade em uma imagem correlaciona-se com alguma característica abstrata. A título de exemplo descontinuidades na profundidade em que os objetos situam-se no mundo real, convertem-se em descontinuidades no domínio do brilho de uma imagem. É possível elencar as correlações normalmente realizadas com descontinuidades que produzem contornos.

- Descontinuidades em profundidade;
- Descontinuidades na orientação de superfícies;
- Mudanças nas propriedades dos materiais;
- Variações na iluminação da cena.

Idealmente, a aplicação de um filtro detector de contornos no domínio da imagem leva a detecção dos limites de um objeto. Em consequência, o domínio em análise para detecção de objetos torna-se menor uma vez que somente a descrição dos contornos encontra-se presente no novo domínio.

Não trivialmente, existem domínios em que após a aplicação de um filtro não formam curvas fechadas. Esse é um problema em os algoritimos de detecção de objetos devem enfrentar quando previamente aplica-se uma filtragem de detecção de contornos.

Outra característica das imagens reais leva a conclusão de que algoritimos de detecção de bordas não são triviais. Devido a compressão, falta de resolução suficiente e outras causas que levam ao esvanecimento de bordas tendem a mesclar contornos, o que resulta mais uma vez em curvas não fechadas e outros problemas que devem ser manipulados de forma não trivial.

2.3.2 *Corner Detection*

A detecção de esquinas são algorítimos que detectam extremidades. Detecção de esquinas geralmente é um algoritimo utilizado em diversas aplicações além da detecção de objetos. Algumas dessas outras aplicações são: na detecção de movimento, registro de imagens, rastreamento de objetos em vídeos, construção de mosaicos, construção de panoramas, modelagem 3D.

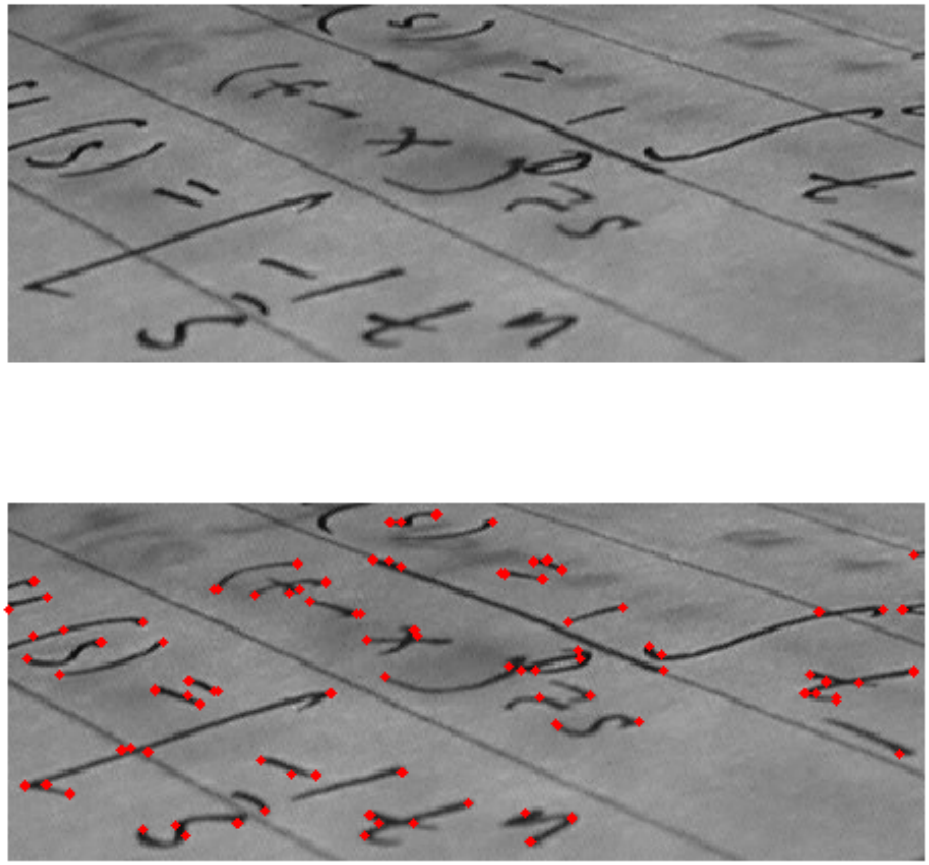


Figura 4 – Exemplo de imagem submetida a um algoritimo de detecção de esquinas.

Fonte: <https://en.wikipedia.org>

Formalmente, uma esquina é a interseção de dois contornos. Um ponto de interesse

no processo de detecção de uma esquina consiste naquele que é possível detectá-lo robustamente. Um ponto de interesse pode ser uma esquina, porém pode ser, por exemplo, um ponto de máxima ou mínima intensidade de brilho local, bem como o fim de uma linha. Na literatura geralmente intercambia-se os termos esquima e ponto de interesse.

A detecção de esquinas é empregado em um dos métodos testados neste trabalho para obtenção de *features* em imagens, o algorítimo FAST.

2.3.3 Critério SUSAN

Antes de apresentar o algoritimo FAST, deve-se apresentar o algoritimo SUSAN. Acrônimo para *smallest univalue segment assimilating nucleous*. É o resultado de uma patente britânica de 1992 intitulada "*Determining the position of edges and corners in images*".(M., 1994).

Para detecção de *features*, o algoritimo SUSAN põe uma máscara circular sobre o pixel a ser testado. Forlmalmente, a região M é a máscara propriamente dita e um pixel na máscara é descrito por $\overrightarrow{m} \in M$. O ponto de análise é representado por $\overrightarrow{m_0}$. Cada pixel na máscara é comparado com a máscara pela função $c(\overrightarrow{m})$ determinada pela equação 2.1.

$$c(\overrightarrow{m}) = e^{-\left(\frac{I(\overrightarrow{m})-I(\overrightarrow{m_0})}{t}\right)^6} \tag{2.1}$$

onde t consiste no limiar de da diferença de brilho entre os pontos. I consiste no brilho do pixel e o expoente determinou-se empiricamente.

A área do SUSAN é dada por:

$$n(M) = \sum_{\overrightarrow{m} \in M} c(\overrightarrow{m}) \tag{2.2}$$

A resposta do operador SUSAN é dada pela equação 2.3.

$$R(M) = \begin{cases} g - n(M) & \text{se } n(M) < g \\ 0 & \text{caso contrário} \end{cases} \tag{2.3}$$

onde g é denominado limiar geométrico.

(SMITH; BRADY, 1997) aprofunda-se detalhadamente no arcabouço matemático do método em questão. Para completa compreensão, recomenda-se a leitura integral e implementação em ambiente de computação numérica o que é descrito no artigo.

algoritimo para detecção de *features* em de esquinas. É derivado do algoritimo AST, um *accelerated segment test* que consiste em um relaxamento de esquinas denominado SUSAN.

2.3.4 *Blob Detection*

Formalmente no campo da visão computacional um *blob* é uma região mais clara ou escura que a sua vizinhança. Trata-se de um método similar aos anteriores no ponto de vista de que consiste em uma das primeiras formas de processamento antes da realização de tarefas mais compilcadas.

Detectores *blob* podem ser agrupados nas seguintes categorias:

- *Matched Filters/Template Matching*: Estes métodos são pouco aplicáveis quando ocorrem deformações na forma dos objetos ocorrem.
- *Watershed Detection*: Este tipo de algorítimos encara as imagens em escala de cinza com um mapa topográfico. Encontra-se os blobs pelo fluxo tomado pela água em uma chuva uniformemente distribuída.
- *Detecção por análise de espaço de escala*: A teoria de espaço de escala ou *scale-space theory* é um arcabouço para processamento de sinais multi-escala desenvolvido pelos estudiosos em visão computacional. É uma teoria formal para lidar com objetos (ou estruturas) de imagens em diferentes escalas. A detecção por espaço de escala é uma metodologia formal e elegante do ponto de vista matemático. Foge ao escopo deste trabalho a apresentação formal dos conceitos que envolvem a detecção por análise por espaço de escala.
- *Tensor de cores*: Hoje em dia, muitos algoritimos para detecção de *features* utilizam o domínio da luminância (brilho, contraste e relacionados). A utilização de tensor de cores é uma alternativa que leva a análise ao domínio de cores de uma imagem.

2.3.5 *Ridge Detection*

Em visão computacional. cumes (*ridges*), refere-se a um lugar geométrico no domínio de uma imagem em que encontra-se um máximo ou mínimo local. Formalmente há uma conexão intima entre a detecção de *edges* e a detecção de *ridges*

2.3.6 *Hough Transform*

A transformada Hough consiste em uma técnica de extração de *features* com o propósito de encontrar imperfeições em instâncias de objetos com certa classe de forma através de um procedimento de votação. Este procedimento de votação é levado a cabo através de espaços paramétricos, dos quais, candidatos ao objeto são obtidos como um máximo local denominado espaço acumulador que são construídos explicitamente através do algorítimo para computação da transformada Hough.

Inicialmente a transformada Hough tinha como objetivo a identificação de linhas em imagens, mais tarde, o método foi extendido a fim de identificar formas arbitrárias.

2.3.7 *Structure Tensor*

Tensores de estrutura ou *structure tensor* é uma matriz derivada do gradiente de uma função. Esta matriz sumariza direções predominantes do gradiente em uma vizinhança de um ponto. Este arcabouço matemático é a base para a análise em espaço de escala e é peça fundamental em algorítimos de detecção de esquinas, pontos de interesse e rastreamento de *features*.

2.4 SIFT - *scale-invariantfeature transform*

SIFT consiste em um acrônimo para *scale-invariantfeature transform*. É um algoritimo para detectar e descrever *features* locais em uma imagem.

Qualquer objeto em uma imagem pode ser submetido à extração de pontos de interesse de forma a realização de uma descrição de artefato. A descrição extraída de uma imagem modelo pode ser utilizada para detecção de objetos em uma cena contendo inúmeros outros objetos. Uma detecção robusta deve garantir que os *features* extraídos da imagem de referência possam ser detectados sob mudanças de iluminação, ruído e de escala. Estes pontos geralmente podem ser obtidos em regiões de alto contraste na imagem de referência como esquinas.

O algorítimo SIFT por sua vez garante que variações na iluminação, escala, orientação e variações parciais por distorções afins não influenciem na detecção dos objetos.

O méotodo SIFT é um algorítimo composto das seguintes partes:

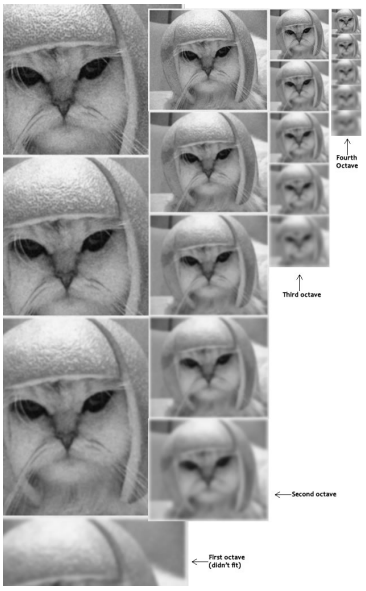
Construção de um espaço de escala É a preparação inicial, gera-se um espaço de escala a fim de garantir invariância a escala.

Aproximação LoG Realiza-se a operação da de laplaciano de uma gaussiana sobre o domínio transformado em espaço de escala a fim de melhorar o custo computacional.

Desfaz-se de pontos de interesse inexpressivos Esquinas e locais de baixo constraste são descartados pois são ineficazes quando utilizados para realizar a operação de correspondência.

Geração de *features* SIFT Finalmente, com a invariância a transformação a fim e escala, uma maior representação é gerada. Isto ajuda a garantir unicidade dos *features*.

(a) Espaço de Escala Comum



(b) A subfigure

Figura 5 – Escala de Espaço do Método SIFT

Fonte:

Após a execução do algorítimo SIFT pontos de interesse ou *keypoints* são fornecidos. Posteriormente deve-se utilizar algum algoritimo para realizar casamento entre os *features* gerados de outra imagem a fim de realizar a detecção do objeto.

2.4.1 Construindo um espaço de escala

Basicamente a construção de um espaço de escala consiste em borrar a imagem original progressivamente utilizando uma função de esvaneicimento gaussiano. Segundo diversas premissas, a utilização de uma gaussiana para tal fim corresponde a uma metodolgia bastante robusta.

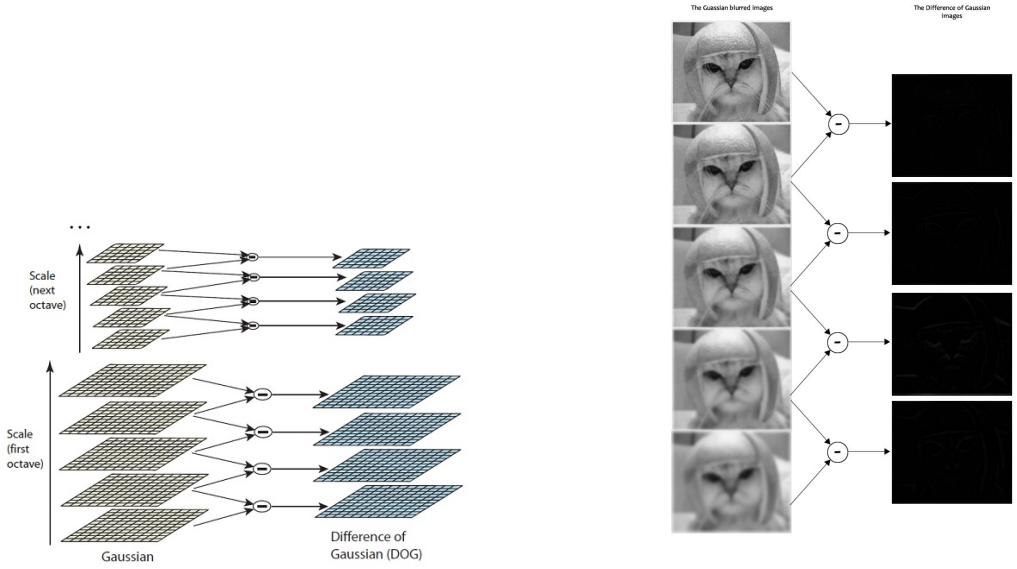
O algorítimo SIFT, por sua vez, além de borrar imagens, as diminui pela metade até a quarta oitava.

O número de oitavas e escalas é derivado do tamanho original da imagem. Enquanto enquanto programa-se o método SIFT, você deve decidir quantas oitavas você quer gerar a partir da imagem original para construção dos *features*.

A primeira oitava da imagem é o dobro do tamanho da original. Realiza-se uma operação de suavização gaussiana.

2.4.2 Laplaciano da Gaussiana

A operação LoG funciona da seguinte forma:



(a) Espaço de Escala Comum (b) A subfigure

Figura 6 – Computação aproximada do procedimento

Fonte:

1. Aplica-se uma suavização gaussiana sobre o domínio da imagem.

2. Calcula-se a derivada de segunda ordem sobre o domínio da imagem.

3. Aplica-se uma suavização gaussiana sobre o domínio da derivada de segunda ordem do domínio da imagem.

O custo computacional destas operações é elevado, portanto, utiliza-se um algorítimo de aproximação para este procedimento: a diferença entre suavizações consecutivas em cada oitava. O algoritimo é ilustrado pela figura

2.4.3 Encontrando pontos de interesse

Para encontrar os pontos de interesse, o algoritimo SIFT utiliza dois passos:

1. Localizar pontos de máximo e mínimo nas imagens geradas pela diferença gaussiana.

2. Localizar máximo e mínimos subpixels.

Encontrar máximos e mínimos locais é uma tarefa simples, iterativa e facilmente paralelizavel. Basicamente em um raio em torno de um pixel, determina-se se aquele pixel é um máximo local.

Encontrar valores de máximos subpixels também é uma tarefa simples e é realizada através da interpolação de segunda ordem da região ao redor do pixel. A equação 2.4 mostra a expressão utilizada para tal finalidade.

$$D(x) = D + \frac{\partial D^7}{\partial x} + \frac{1}{2}x^T \frac{\partial^2 D}{\partial x^2} x \tag{2.4}$$

2.4.4 Removendo *features* de baixo contraste

Utiliza-se uma função de limiar de contraste para determinar se o pixel em questão é utilizável. Novamente, para encontrar pontos chave ao nível subpixel deve-se utilizar a função 2.4.

2.4.5 Removendo esquinas

A ideia baseia-se em calcular dois gradientes perpendiculares entre si baseado na região ao redor do ponto de interesse. É possível deduzir algumas propriedades do ponto de interesse a partir dessa operação.

1. O ponto de interesse faz parte de uma **região plana** se ambos os gradientes são baixos.
2. O ponto de interesse faz parte de uma **região de borda** se apenas um dos gradientes for alto.
3. O ponto de interesse faz parte de uma **região de canto** se os dois gradientes forem altos.

2.4.6 Orientação dos pontos de interesse

Após todos os passos anteriores, obtemos um ponto chave representativo com respeito a imagem base para identificação de um objeto. Sabe-se qual a escala do objeto, uma vez que a escala é a mesma em que ocorreu a detecção. Portanto, existe uma invariância a escala pela propagação dos pontos chave através de todas as escalas geradas pelo algoritimo.

Computa-se todas as orientações através das seguintes fórmulas:

$$m(x,y) = \sqrt{[L(x+1,y) - L(x-1,y)]^2 + [L(x,y+1) - L(x,y-1)]^2} \tag{2.5}$$

$$\theta(x,y) = \tan^{-1}\left[\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right] \tag{2.6}$$

onde $m(x, y)$ refere-se a magnitude no ponto (x, y) e $\theta(x, y)$ refere-se ao ângulo de fase.

A magnitude da orientação é calculada para todos os pontos ao redor do ponto de interesse. Um histograma é computado para divisões de ângulos de 10 em 10 graus. A orientação do ponto chave é então computada como o agrupamento com maior magnitude. Pontos-chame secundários são gerados para agrupamentos com predominância superior a 80% no histograma.

2.4.7 Gerando um artefato

A geração de artefato consiste em construir uma identificação única que armazene as características obtidas no ponto de interesse de forma que seja simples de calcular e comparar com outros *features*. Para isso, o método SIFT define uma janela de 16x16 pontos ao redor do ponto chave que é quebrado em 16 janelas de 4x4 pontos. Para cada janela 4x4 o gradiente, a magnitude e orientação são calculadas. Da mesma forma que na seção anterior, calcula-se

2.5 SURF - *Speeded Up Robust Features*

O método SIFT anteriormente explanado é um método consideravelmente lento. Em 2006, três pesquisadores, Bay, H., Tuytelaars, T. e van Gool, L, publicaram um paper com o título "*SURF: Speeded Up Robust Features*". O método sugere uma alternativa ao SIFT que provê resultados melhores e mais rápidos.

No SIFT, a aproximação DoG para encontrar o espaço de escala tem um custo computacional elevado. O método SURF vai além na aproximação, neste caso, ele utiliza um filtro *box* para realizar operação análoga. A imagem abaixo mostra a demostração desta aproximação. Uma das grandes vantagens da utilização do filtro é que ele pode ser paralelizável e calculado facilmente.

Um maior detalhamento do método pode ser obtido no trabalho original que explica com clareza e riqueza de detalhes a implementação do método. Infelizmente o método SURF é protegido por patente, mas livre para uso acadêmico.

2.6 ORB - *Oriented FAST and Rotated BRIEF*

O método ORB é uma técnica alternativa às patenteadas desenvolvida pelo time de engenheiros que constroem o OpenCV. O algoritimo foi concebido por Ethan Rublee, Vicent Rabaud, Kurt Konolige e Gary R. Bradski no *paper* "*ORB: An efficient alternative to SIFT or SURF*"em 2011. Como o titulo do trabalho sugere, é uma boa alternativa para

o SIFT e o SURF em relação ao custo computacional, equiparando-se em desempenho e com custo mais accessível, uma vez que o método é livre de patentes.

O algoritimo é basicamente uma fusão de dois algoritimos, o FAST, que consiste em um detector de *keypoints* e o BRIEF, um algoritimo para encontrar *descritores*. Sabe-se, porém, que o método FAST não é invariante a rotação. Os autores do método ORB, para mitigar o problema, propuseram uma alteração no FAST de forma a mitigar o problema.

2.6.1 FAST - *Features from Accelerated Segment Test*

FAST é um acrônimo para *Features from Accelerated Segment Test*. É um algoritimo de detecção de *corners* desenvolvido por Edward Rosten e Tom Drommond em 2006 no trabalho *Machine Learning fro High-Speed Corner Detection*(ROSTEN; PORTER; DRUMMOND, 2010). Este detector provê ampla estabilidade.

O algoritimo em questão é uma derivação do algoritimo AST (*Accelerated Segment test*). AST é uma modificacao do critério SUSAN. AST difere do criterio SUSAN de forma que todos os pixels são avaliados no segundo enquanto que o AST só avalia os pixels em um círculo Bresenham, que são os necessários para rasterização de um círculo.

O algoritimo FAST considera uma *feature* em um determinado pixel p se n pixels contiguos são todos mais brilhantes que o núcleo por pelo menos t ou todos mais escuros que o núcleo por t. Raios mais largos em torno do pixel de teste tendem a ser mais ruidosos e resistentes ao desvanecimento enquanto que são computacionalmente mais caros uma vez que mais pixels deverão ser comparados a fim de verificar o atendimento ao criterio estabelecido para ser considerado uma feature.

O algoritimo FAST é considerado o mais eficiente algoritimo de detecção de *corners* já desenvolvido até hoje.

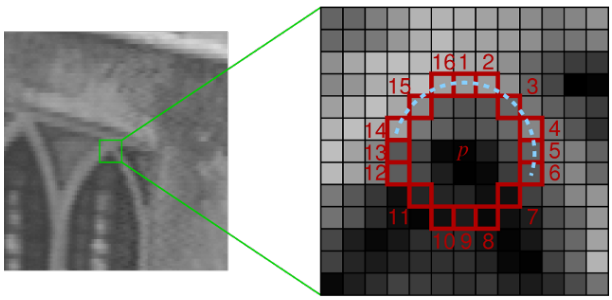


Figura 7 – Exemplo de uma circunferência de Bresenham(CONTRIBUTORS, 2018) utilizada para determinação de uma esquina utilizando o algoritimo FAST

Fonte: <www.edwardrosten.com>

2.6.2 BRIEF - *Binary Robust Independent Elementary Features*

Em geral, após a extração dos descritores de uma imagem, necessita-se compará-los. O algoritimo BRIEF encarrega-se de comparar a similaridade entre *keypoints* de duas imagens.

2.7 Biblioteca OpenCV

2.7.1 Recursos Básicos

A biblioteca OpenCV é um conjunto de algoritimos reunidos de forma ordenada e gerenciada a fim de reunir uma grande quantidade de rotinas relacionadas ao tema de visão computacional. É um projeto Open Source veiculado pela Intel, Willow Garage, Itseez. Sua primeira versão data de meados dos anos 2000. e atualmente, quase 18 anos depois, está na versão 3.4.0. É uma biblioteca escrita em C++ com interface para diversas linguagens, oficiais ou não. É multi-plataforma e pode ser utilizada em diversos sistemas operacionais e arquiteturas de processamento.

É uma biblioteca desenvolvida com foco em eficiência computacional com foco em aplicações de tempo real e quase sempre lança mão de rotinas otimizadas em hardware para processamento de imagens. Muito esforço tem sido feito para que a maioria dos algoritimos suportem as mais diversas plataformas de aceleração de hardware.

2.7.2 *Feature Detection Module*

O módulo do OpenCV que engloba as funcionalidades necessarias para realizar detecção de pontos chave e extração de descritores para realização de detecção de *features* de uma imagem é denominado `features2d`.

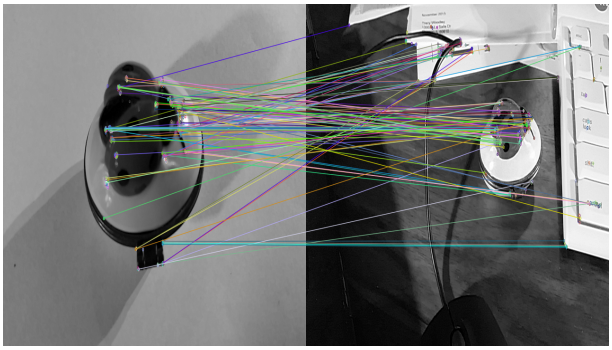


Figura 8 – Exemplo de resultado de*feature* matching utilizando a biblioteca OpenCV.

Fonte: <https://opencv.org>

2.8 Arquitetura Embarcada

Neste projeto, experimentou-se o desempenho de uma plataforma embarcada desenvolvida pela Raspberry Pi Foundation, denominada Raspberry Pi 3 Modelo B com vistas a verificar se os algoritimos de detecção de objetos supra-citados podem ser estendidos ao hardware.

2.8.1 Raspberry Pi

A Raspberry Pì é um computador básico de baixo custo que originalmente foi desenvolvido para ajudar crianças interessadas por computadores a conseguirem desenvolver sua curiosidade. Na primeira versão, a Raspberry Pi era um computador de placa única que contem as seguintes caracteristicas:

- HDMI;
- USB 2.0;
- Video Composto;
- Audio Analógico;
- Fonte de Alimentação;
- Acesso a Internet;
- Cartão SD.

O computador roda inteiramente software open-source, apesar de o hardware não o ser. Os modelos geralmente integram um processador ARMv7 licenciado pela Broadcom. Aos propositos iniciais, essas caracteristicas são suficientes a qualquer uma criança com curiosidade aguçada.

2.8.1.1 Raspberry Pi 3

O hardware utilizado para experimentação consiste em uma Raspberry Pi modelo 3. As suas caracteristicas especificas podem ser listadas conforme segue:

- SoC: Broadcom BCM2837B0;
- Processador: 1.4 GHz 64/32-bit quad-core ARM Cortex-A53;
- Memoria: 1 GB LPDDR2 RAM at 900 MHz;
- Armazenamento: MicroSDHC;

- Gráficos: Broadcom VideoCore IV 300 MHz/400 MHz, HDMI.
- Interface: Barramento GPIO com SPI e I2C, Terminal para Câmera e Display, USB 2.0;
- Rede: WiFi, Bluetooth e Ethernet.

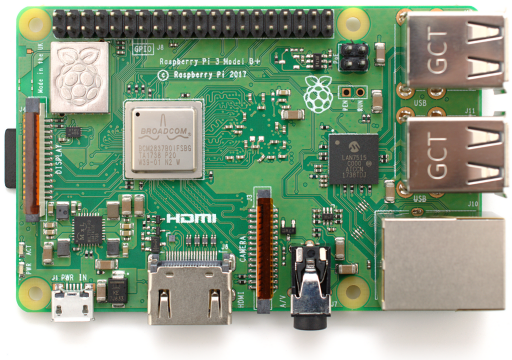


Figura 9 – Raspberry Pi 3 Model B

Fonte: <https://en.wikipedia.org>

2.8.2 Raspbian GNU Linux

O sistema operacional utilizado para realizar testes foi o Raspbian, que foi especialmente desenvolvido para a plataforma em questão. É uma distribuição Linux baseada na bem consolidada distribuição denominada Debian.

O sistema operaional herda muitas características de sua família de sistemas operacionais, como os gerenciadores de pacotes. É uma distribuição simples com alguns aplicativos pra o gerenciamento basico do sistema, no caso da versão para console serial, mas possui uma versão com vários aplicativos, incuindo um gerenciador de desktop de baixo consumo de recursos denominado MATE Desktop.

Implementações do método SIFT, SURF e ORB estão disponíveis na biblioteca. Sendo que o SIFT e o SURF foi, recentemente movida para a versão extra `xfeatures2d`. A ideia é que, para evitar problemas legais, os algoritimos protegidos por patentes sejam separados daqueles que não são.

2.9 Tecnologias Adicionais

2.9.1 HTTP Server

Para realização dos experimentos, construiu-se um serviço web para processamento de requisições externas com a tecnologia Node.js. Um framework para criação de serviços

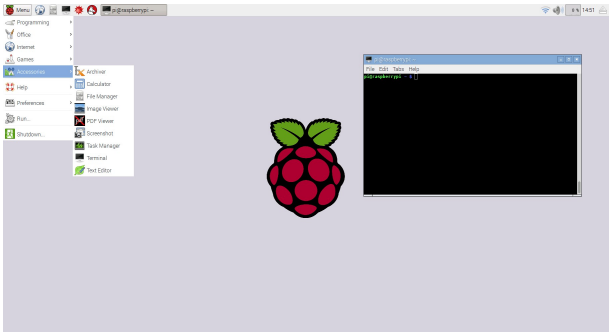


Figura 10 – Screenshot do sistema operacional Raspbian utilizando gerenciador de desktop MATE.

Fonte: <https://en.wikipedia.org>

baseados em web utilizando JavaScript.

2.9.2 Node.JS

Node.js é um abiente open-source, cross-platform baseado em Javascript para execução de código de servidor. Javascript, tradidionalmente, foi utilizado no lado do cliente embarcado em páginas HTML para garantir conteúdo dinâmico. Node.js permite que aplicações do lado do servidor possam também ser escritas de forma a servir conteúdo, sendo assim, unifica-se em apenas um paradigma de programação aplicações que rodam tanto como servido quanto consumindo conteúdo na internet.

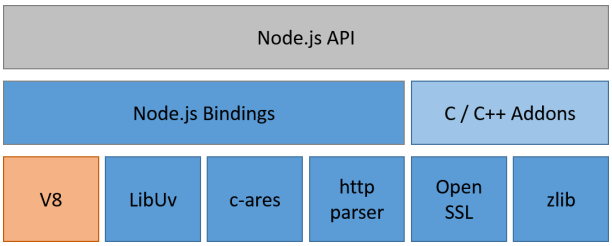


Figura 11 – Arquitetura básica do Node.js.

Fonte: <stackoverflow.com>

Node.js possui várias opções de interface com a biblioteca OpenCV, escrita em C++. Todas disponíveis sob licensa open-source. Experimentar o desempenho destas soluções é importante, uma vez que a construção de serviços para processamento de dados é fundamental em aplicações envolvendo internet.

2.9.2.1 ExpressJS

ExpressJS ou simplesmente Express é um framework de aplicações para Node.js liberado sob licensa open-source. Foi desenvolvido para construir aplicações, serviços web e APIs. É a solução padrão para servidores HTTP em Node.js. O autor original T. J.

Holowaychuk, descreveu a própria solução como mínima, mas versátil pela facilidade em adicionar plugins.

Utilizou-se o framework por ser padrão na construção de serviços com Node.js. Os experimentos envolvendo esta plataforma devem mostrar o desempenho em um sistema baseado em web típico.

3 Análises e Resultados

3.1 Análise de Desempenho

Para verificar a desempenho das requisições traçou-se um perfil dos tempos de envio de requisições e respostas, tomando como base a ferramenta de perfilação do navegador utilizado, o Google Chrome.

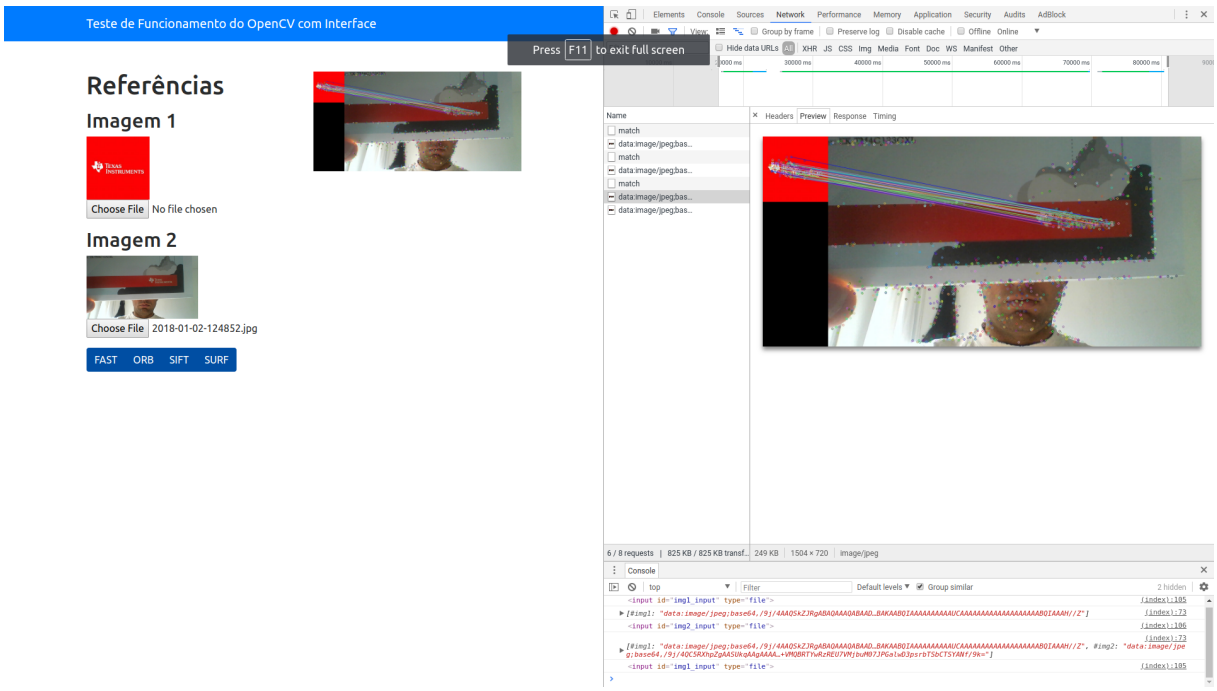


Figura 12 – Captura de tela da aplicação desenvolvida para testes

Fonte: Autoria Própria

Os testes realizados resultaram que para o método ORB, o tempo médio de processamento pelo sistema ficou em torno de 4.20 segundos para uma imagem de referência de 244 pixels por 244 pixels, com o match sendo realizado com uma imagem de 1280 pixel por 720 pixels. A Figura 13 mostra o resultado dos matches obtidos entre as duas técnicas.

Submetendo as mesmas imagens ao método patenteado SIFT, o tempo de processamento elevou-se mais que 10 vezes a um patamar de 45 segundos. Sendo assim, o método torna-se inviável a aplicação que requer tempos de resposta menores. Como previsto na técnica a quantidade de acertos em relação aos pontos de match são semelhantes o que configura a técnica como uma alternativa inviável, já que o desempenho de uma técnica computacionalmente menos custosa obtém um resultado semelhante.

Figura 13 – Feature match utilizando a técnica ORB

Fonte: Autoria Própria

Figura 14 – Feature match utilizando a técnica SIFT

Fonte: Autoria Própria

A técnica SURF, por sua vez, obteve um tempo de processamento de 6.76 segundos. Um pouco maior que a técnica ORB, sendo irrelevantemente mais custoso que a técnica ORB. Assim como a técnica SIFT, o número de acertos de features foi semelhante a da técnica ORB. Portanto, descarta-se a utilização do mesmo, uma vez que a técnica também é protegida por patentes que impedem uso comercial.

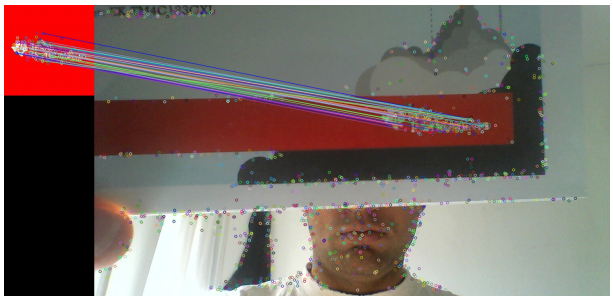


Figura 15 – Feature match utilizando a técnica SURF

Fonte: Autoria Própria

4 Conclusão

4.1 Sugestões para trabalhos futuros

Diante dos resultados obtidos, evidencia-se a superioridade do algoritimo ORB no que refere-se ao desempenho. O método ORB mostrou-se superior em tempo processamento a todos os outros algoritimos apresentando resultados similares no que refere-se a identificação do objeto em analise.

Uma análise mais profunda dos algoritimos a nível de implementação pode garantir ainda uma maior vantagem para alcançar objetivos como tempo real. A utilização de um sóftware em código nativo para a plataforma utilizada pode também ser uma alternativa a implantação de um sistema de visão utilizando uma Raspberry Pi, pois, a linguagem Node.js, pode ser um gargalo por conta das camadas de abstração que implementa em JavaScript.

Referências

CONTRIBUTORS, W. *Midpoint circle algorithm — Wikipedia, The Free Encyclopedia*. 2018. [Online; accessed 18-March-2018]. Disponível em: <https://en.wikipedia.org/w/index.php?title=Midpoint_circle_algorithm&oldid=823441392>. Citado 2 vezes nas páginas 15 e 36.

FREEMAN, W. T.; ROTH, M. *Orientation Histograms for Hand Gesture Recognition*. Cambridge, MA 02139, 1994. Disponível em: <http://www.merl.com/publications/TR94-03/>. Citado na página 24.

M., S. S. *Determining the position of edges and corners in images*. [S.l.]: Google Patents, 1994. Great Bretain Patent 2,272,285. Citado na página 29.

MCCONNELL, R. K. *Method of and apparatus for pattern recognition*. [S.l.]: Google Patents, 1986. US Patent 4,567,610. Citado na página 24.

ROSTEN, E.; PORTER, R.; DRUMMOND, T. Faster and better: A machine learning approach to corner detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, v. 32, p. 105–119, 2010. Disponível em: <http://lanl.arXiv.org/pdf/0810.2434>. Citado na página 36.

SMITH, S. M.; BRADY, J. M. Susan—a new approach to low level image processing. *International Journal of Computer Vision*, v. 23, n. 1, p. 45–78, May 1997. ISSN 1573-1405. Disponível em: <https://doi.org/10.1023/A:1007963824710>. Citado na página 29.

VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. In: IEEE. *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. [S.l.], 2001. v. 1, p. I–I. Citado na página 24.