Felipe Gustavo Bombardelli

# Localização de Robôs Móveis por Aparência Visual

**Curitiba**

**2014**

Felipe Gustavo Bombardelli

# Localização de Robôs Móveis por Aparência Visual

Trabalho de Graduação apresentado como requisito parcial à obtenção do grau de Bacharel em Ciência da Computação pelo Setor de Ciência Exatas da Universidade Federal do Paraná

Universidade Federal do Paraná – UFPR

Departamento de Informática

Orientador: Eduardo Todt

Curitiba

2014

# Resumo

Este trabalho tem como objetivo buscar um método para localizar um robô através de aparência visual, ou seja, a partir de imagens de uma única câmera consiga determinar em qual ambiente ou sala o robô está. Para este proposito foi abordado dois caminhos diferentes, primeiro utilizando pontos de interesse e segundo por regiões estáveis da imagem. Também foi proposto um novo algoritmo de detecção de área estável, o qual obteve bons resultados nos testes realizados. Isto indica que o algoritmo proposto é promissor sendo interessante a pesquisa de aperfeiçoamento e validação do mesmo.

**Palavras-chaves**: localização, robótica, processamento de imagem.

# Sumário

# 1 Introdução

A localização é um problema fundamental para robótica móvel, devido à grande importância de determinar a posição do robô no espaço. Este dado junto com o mapa do ambiente são informações básicas para a mobilidade do robô de um local A para um local B.

Atualmente, para lugares externos, pode-se usar um dispositivo como *Global Positioning System*(GPS), que permite obter facilmente as coordenadas do robô. Contudo, para ambiente internos, dificilmente o receptor consegue captar o sinal do GPS e portanto é necessário utilizar outros métodos para obter a posição. Logo, o robô necessita de algum dispositivo para captar informações do ambiente, como sonares, sensores a laser e odometria. Neste trabalho utilizou-se exclusivamente o uso de uma única câmera, pois o interesse é a aparência visual. Portanto este projeto visa obter um método eficaz para elaborar uma base de dados, a qual armazene as características de um conjunto de ambientes, e possibilite que um robô móvel, com esta base de dados e uma câmera seja capaz de determinar em qual ambiente se localiza.

Esta informação é importante pois possibilita ao robô buscar informações adicionais sobre este ambiente em uma outra base de dados, e também este método pode ser utilizado, por exemplo, na união de mapas construídos por diversos robôs através da aparência visual, já que no problema da união um dos grandes desafios é saber quais das partes destes mapas representam o mesmo lugar.

Este trabalho está estruturado da seguinte forma. Um capítulo sobre a revisão da literatura, o qual dará uma visão geral de todos os algoritmos utilizados, ou seja, os algoritmos HARRIS, SIFT, SURF e FAST. Em seguida o capítulo sobre a implementação, o qual abordará inicialmente como foi elaborado a base de dados dos ambientes e como foi realizado a classificação de um vídeo. Logo após é descrito como funciona o novo detector de pontos proposto por este trabalho e for final os resultados de cada algoritmo, seguido pela conclusão.

# 2 Revisão da Literatura

Ulrich e Nourbakhsh(1) em 2000 apresentaram um trabalho com o foco na localização de robô por aparência visual e sua principal contribuição foi a utilização da distribuição de cores para descrever uma imagem. Este descritor é formado por 3 histogramas no formato HSV (Matiz, Saturação e Valor), já que este formato apresenta características melhores que o tradicional RGB(Vermelho, Verde e Azul). Porém muitas imagens de lugares distintos tem histogramas parecidos, deixando assim a correspondência entre imagens bastante imprecisa, com uma taxa de acerto de apenas 36,7% como descrito no trabalho de Chao, Yucheng e Tieniu(2) em 2003.

Então, esses mesmos autores abordaram o problema de precisão deste descritor e adicionaram mais algumas métricas da imagem, como quantidade de bordas, gradiente de magnitude, textura da imagem como descrito por Engelson(3), e rank de pixeis, baseado no contraste. Com essas melhorias, eles aumentaram a taxa de acerto para 82%, entretanto, elevou-se o custo computacional.

Em 1999 o algoritmo Scale-Invariant Feature Transform(SIFT) foi publicado (4) por David Lowe e tornou-se popular após 2004 com sua segunda publicação(5) e representou uma revolução no processamento de imagens. Este algoritmo detecta vários pontos da imagem com grande importância, usando como base a ideia de detecção de cantos proposto por Harris e Stephens(6). Contudo a grande revolução do SIFT foi a elaboração de um descritor capaz de comparar dois pontos de interesses através de um descritor invariante a escala, rotação, iluminação e pouco afetado pelo ruído. Logo, pelos ótimos resultados obtidos por este algoritmo, mudou-se a técnica de comparação de imagens de histogramas de cores para usar este novo método, o qual utiliza histograma de gradiente.

Entretanto o SIFT tem alguns problemas, principalmente o alto custo computacional, o qual foi abordado por Ke e Sukthankar em 2004(7) e desenvolveram o PCA-Sift, o qual apresenta um descritor menor que o descritor do SIFT e portanto há uma diminuição do custo computacional para extrair e comparar os descritores. Já Tamimi em seu trabalho em 2006(8) encontraram mecanismos para diminuir o custo da detecção do ponto de interesse, já que estes se localizam normalmente em regiões de alto contraste.

Outro problema, diz a respeito à correspondência dos pontos encontrados com os pontos guardados no banco de dados, já que muitas vezes o algoritmo relaciona pontos errados. Esse problema foi abordado por Zingaretti em 2007(9), que elaborou um modo de agrupar vários pontos próximos e utiliza as informações espaciais dos pontos para melhorar a corretude do algoritmo.

## 2.1 Detector Harris

O algoritmo proposto por Harris(6) utiliza como base a expansão de Taylor, e visa como objetivo uma fórmula, que recebe uma imagem preta e branca e uma coordenada de um ponto, e retorne um número, que se negativo corresponde a um ponto de borda, se positivo a um ponto de superfície plana ou canto, dependendo da magnitude do valor. Neste trabalho este número será representado pela letra $M_c$.

O calculo do $M_c$ tem como base os autovalores da matriz A, que é composto segundo a equação 2.1 e seus autovalores representados por $\alpha$ e $\beta$.

$$A = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{2.1}$$

Sendo que, $D_{xx} = D_x^2$, $D_{yy} = D_y^2$ e $D_{xy} = D_x D_y$ e, por final, $D_x$ é a derivada do ponto no eixo X e $D_y$ é a derivada do ponto no eixo Y. Também importante considerar, que na implementação do algoritmo Harris no OpenCv as derivadas são calculadas através do método Sobel(10).

Assim dependendo dos valores de $\alpha$ e $\beta$, o ponto analisado pode ser classificado como região de borda, região de canto ou região de superfície como mostrado na figura 1.

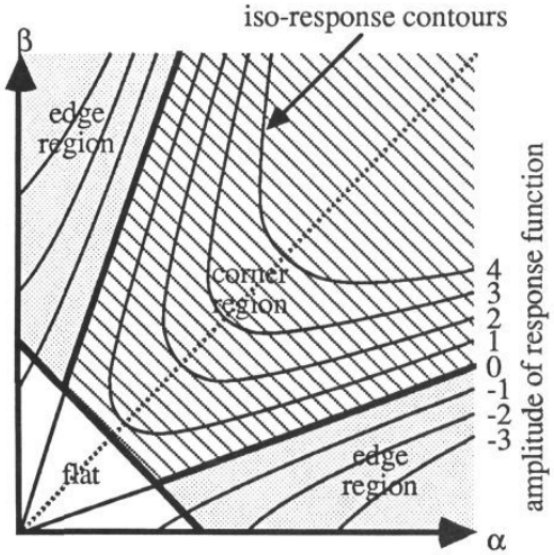


Figura 1: Figura que mostra a classificação de um ponto entre região de borda, superfície e canto dependendo dos valores de $\alpha$ e $\beta$. Figura do trabalho(6).

Portanto o calculo do $M_c$ é realizado através da fórmula 2.2, onde A é a matriz mostrada na equação 2.1. Lembrando que para cada ponto da imagem é calculado seu respectivo $M_c$

$$M_c = Determinante(A) - \kappa Trace(A)^2 \tag{2.2}$$

Onde,

$$Trace(A) = \alpha + \beta = D_{xx} + D_{yy} \tag{2.3}$$

$$Determinante(A) = \alpha\beta = D_{xx}D_{yy} - D_{xy}^2 \tag{2.4}$$

Portanto temos que

$$M_c = (D_{xx}D_{yy} - D_{xy}^2) - \kappa(D_{xx} + D_{yy})^2 \tag{2.5}$$

Na figura 2 pode-se ver o algoritmo em um fluxograma baseado na implementação da biblioteca OpenCv, onde consta a primeira etapa em aplicar o filtro Sobel pela imagem, logo após calcular o $D_{xx}^2$, $D_{yy}^2$ e $D_{xy}$ para cada ponto. Portanto o resultado da 2º etapa são 3 matrizes que correspondem a cada derivada, entretanto na figura 2 essas 3 matrizes estão na mesma imagem, mas em canais diferentes de cores. Então na 3º etapa é realizado um filtro o qual calcula a média do ponto com os seus vizinhos e por final, na ultima etapa, aplica a equação 2.5 para cada ponto da imagem.

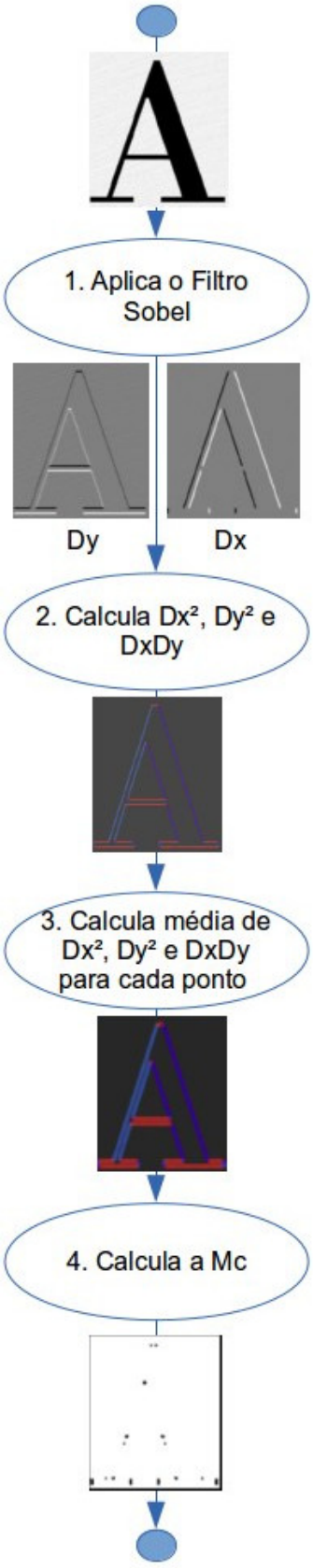


Figura 2: Diagrama do Algoritmo Harris baseado no código implementado no OpenCv versão 2.4.9. Figura do autor.

A figura 3 mostra os resultado do algoritmo Harris para as imagens da cozinha, saguão e o laboratório VRI. Lembrando que o algoritmo Harris não considera a escala, portanto todos os seus pontos tem o mesmo tamanho.

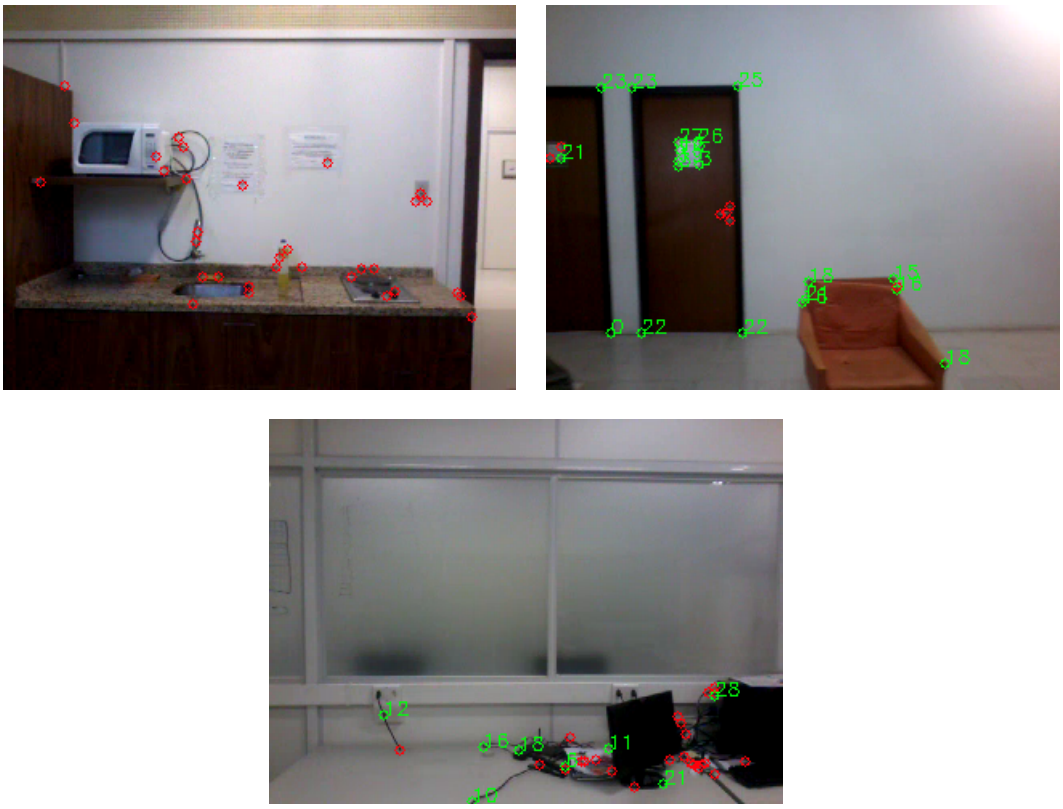

Figura 3: As figuras mostram os resultado do algoritmo HARRIS. Figura do autor.

## 2.2 Detector SIFT

Inicialmente o algoritmo SIFT foi estruturado em 4 partes principais pelo David Lowe em seu trabalho(5). A primeira etapa é a construção da *Scale-space extrema detection*, segunda a localização dos pontos de interesse, terceira a atribuição da orientação dos pontos chaves e, por final, a extração dos descritores para cada ponto de interesse. Contudo a implementação da biblioteca OpenCv segue a tendência de dividir essa classe de algoritmo em duas etapas, a primeira detectar os pontos de interesse e a segunda extrair os descritores. Isso mantêm a liberdade de escolher um algoritmo para detectar e outro algoritmo para descrever, dependendo do problema abordado.

A detecção dos pontos através do algoritmo SIFT recebe uma imagem em tons de cinza e segue os seguintes passos: primeiro a construção da piramide gaussiana, seguido pela subtração das imagens, supressão de não máximos e por final a eliminação dos pontos que estão sobre retas.

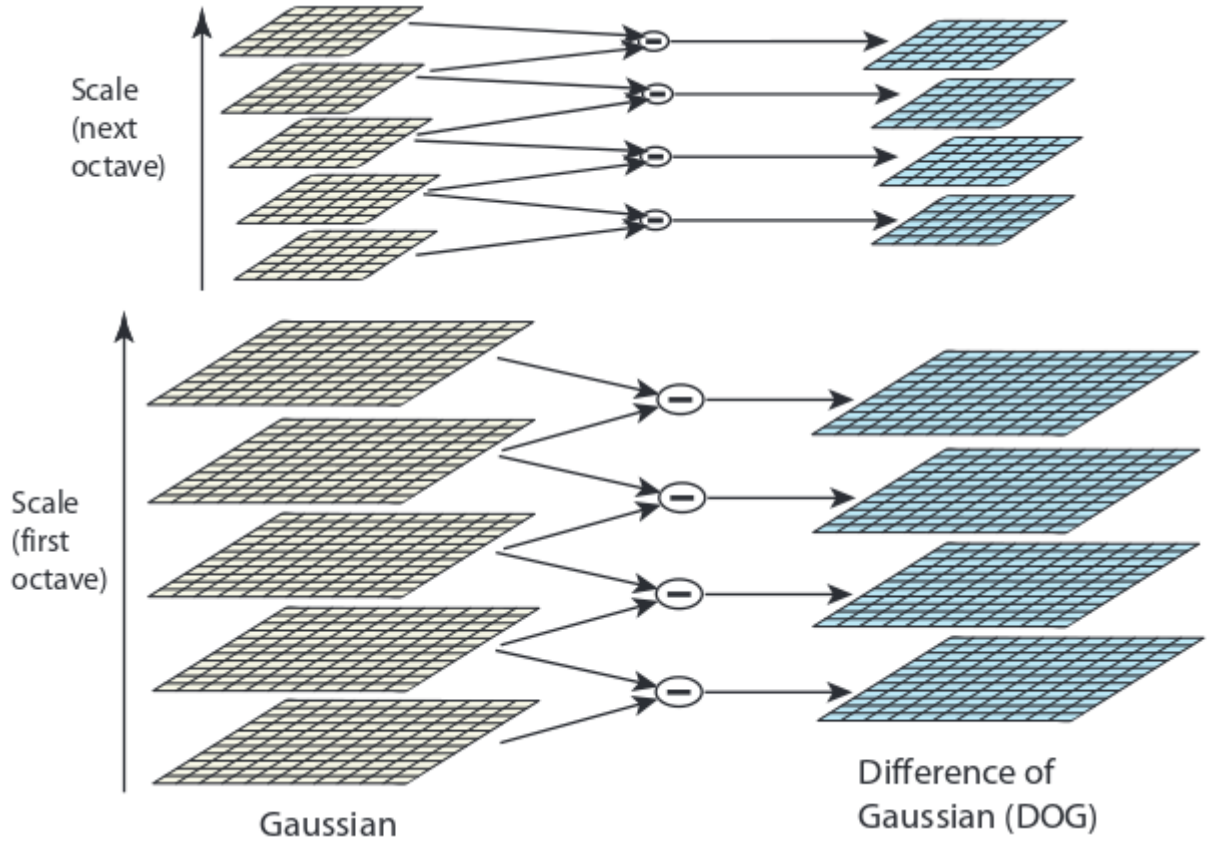


Figura 4: A esquerda mostra a piramide gaussiana com 2 oitavas e 5 escalas cada uma, seguido pela sua subtração e a formação da diferença gaussiana a direita, formada por 2 oitavas com 4 escalas. Figura retirada do trabalho(5).

A piramide gaussiana é composta por dois conceitos importantes, a oitava e a escala, que podem ser visto na figura 4. Cada oitava reduz pela metade o tamanho da imagem, ou seja, na primeira oitava o tamanho da imagem é idêntico a imagem de entrada, já na segunda oitava a imagem é reduzida pela metade e na terceira para $\frac{1}{4}$ da imagem e assim por diante. Cada oitava é composta por várias escalas, as quais são formadas por diferentes filtros gaussianos.

Após a construção da piramide gaussiana é obtido a diferença gaussiana de todas as escalas, que nada mais é do que a subtração da escala posterior com a escala anterior, como mostrado na figura 4. Assim o próximo passo é suprimir os pontos não máximos.

A supressão de não máximos irá percorrer todas as escalas da diferença gaussiana, exceto as que estão nos extremos das oitavas. Pois os extremos não tem uma das escalas, já que como mostrado na figura 5 o valor do ponto central é comparado com os seus 8 vizinhos da mesma escala e seus 9 vizinhos da escala superior e inferior. Assim se o ponto for maior que todos os seus vizinhos, ele é adicionado a lista de pontos de interesse.

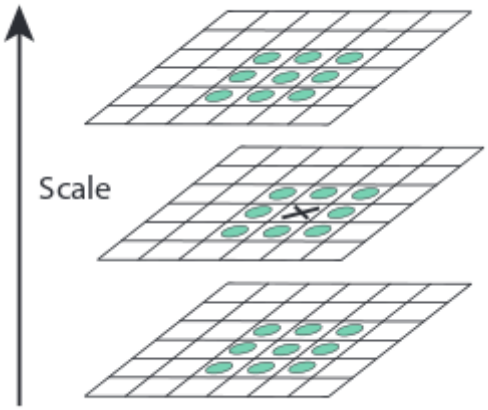


Figura 5: Figura mostra o ponto centrar marcado por um X, o qual é comparado com seus 8 vizinhos da mesma escala e seus 9 vizinhos da escala superior e inferior. Figura retirada do trabalho(5).

Ao final existem muitos pontos que estão sobre retas e não necessariamente sobre os cantos. Estes pontos são ruins, pois surge diversos pontos sobre a mesma reta além que seus descritores são muito parecidos, portanto aplica-se basicamente mesma a fórmula do algoritmo HARRIS, e correponde a equação 2.1. Se o resultado da equação for menor que uma constante, então ele é descartado. Por final, na figura 6 tem alguns resultados do algoritmo SIFT.

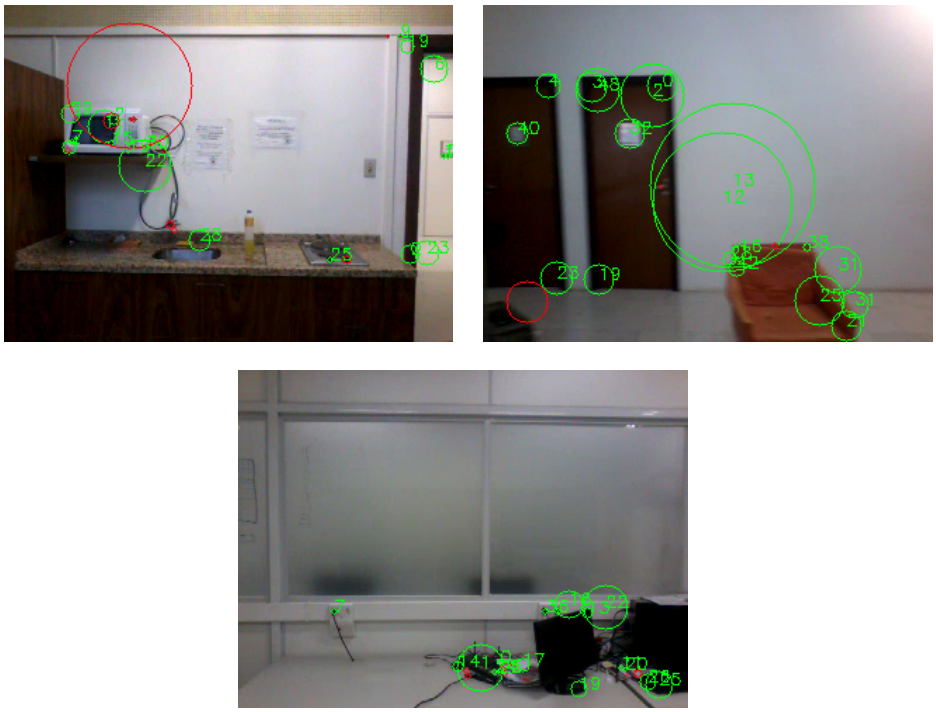

Figura 6: As figuras mostram alguns resultados do algoritmo SIFT. Figura do autor.

## 2.3 Detector SURF

Herbert Bay, quando trabalhou na Computer Vision Laboratory ETH de Zurique em 2006, desenvolveu o algoritmo SURF(11) com mesmo objetivo do SIFT, detectar e descrever os pontos de interesse, porém com um custo computacional inferior.

O algoritmo SURF tem um outro método para formar a diferença gaussiana utilizada no SIFT, entretanto com um custo menor. Por exemplo, no SIFT deve-se aplicar 5 vezes o filtro gaussiano sobre uma imagem para formar uma oitava com 5 escalas, logo após subtraí-las para resultar uma diferença gaussiana de uma oitava com 4 escalas, ou seja, o algoritmo deve percorrer no mínimo 9 vezes a imagem para montar a diferença gaussiana. Já o SURF, como mostrado na figura 9, integra a imagem e monta as 4 escalas a partir de seu método. Portanto percorre apenas 5 vezes a imagem para formar a mesma diferença gaussiana.

Este método é baseado na matriz hessiana assim como o HARRIS, logo temos:

$$A = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{2.6}$$

$$Determinante(A) = D_{xx}D_{yy} - (0.9D_{xy})^2 \tag{2.7}$$

Diferemente do algoritmo HARRIS, o qual usa o filtro Sobel para calcular o $D_{xx}$,$D_{yy}$ e $D_{xy}$, o SURF utiliza *box filter*, como mostrado na figura 7. Então as escalas são formadas usando inicialmente filtro de tamanho 9x9, depois 15x15, 21x21 e assim por diante. Importante lembrar, que o SURF aproxima o resultado do *boxfilter* de tamanho 9x9 com o resultado do filtro gaussiano usando $\sigma = 1.2$, portanto aparece na equação da determinante a constante de 0,9 definida pelo Herbert(11).

Contudo existe um problema, se para cada ponto precisar percorrer toda a região para obter a soma de seus pixeis, então o algoritmo será mais custoso, que o SIFT.

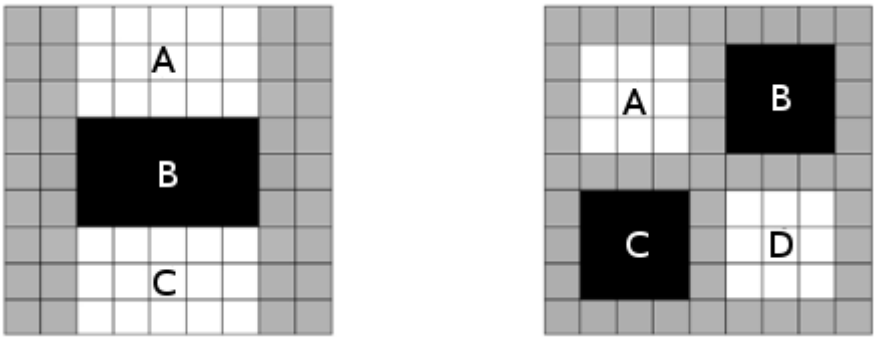


Figura 7: A figura a esquerda mostra o box filter de tamanho 9x9, para calcular o $D_{yy}$, lembrando que $D_{yy} = A - 2B + C$, sendo A,B e C a soma dos pixeis de sua respectiva região. A figura a direita mostra o box filter de tamanho 9x9, para calcular o $D_{xy}$, lembrando que $D_{xy} = A - B - C + D$, sendo A,B,C e D a soma dos pixeis de sua respectiva região. A região cinza não é considerado. Figura adaptada do trabalho(11).

Este problema é resolvido utilizando a integral da imagem, que tem a propriedade de obter a soma de todos os pixeis de uma região retangular de qualquer tamanho da imagem entrada com uma simples formula, como mostrado na figura 8;

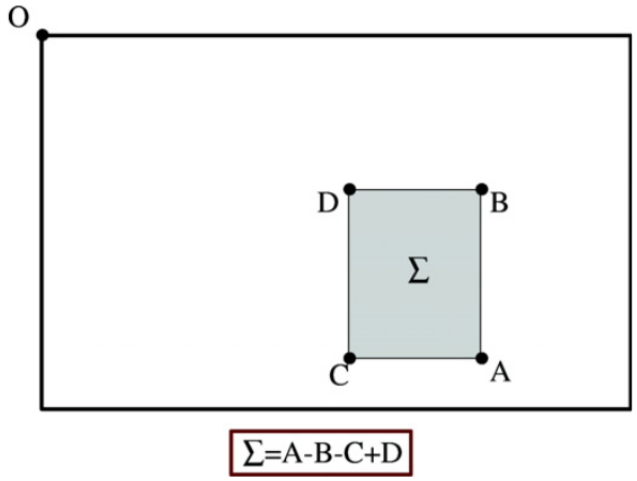


Figura 8: Figura mostra uma subregião formada pelos pontos A,B,C,D sobre uma imagem, a qual tem o ponto de origem marcado pela letra O. Já o símbolo Σ representa a soma de todos os pixeis da subregião. Logo abaixo tem a equação fornecida pela propriedade da integral de uma imagem, onde A,B,C e D são os valores dos pontos na imagem integrada. Figura retirada do trabalho(11).

Na última etapa, a detecção dos pontos, terá como entrada todas as escalas das oitavas e buscará pelos máximos locais. Essa busca utiliza a mesma técnica do algoritmo

SIFT, ou seja, o ponto será comparado com os seus oitos vizinhos da mesma escala e também com seus 9 vizinhos da escala anterior e posterior, como mostrado na figura 5. Se o valor desse ponto for maior que todos, então é colocado como ponto de interesse.

Pode-se observar o fluxograma do algoritmo SURF na figura 9, o qual tem a integração da imagem na primeira etapa e o elaboração de cada filtro na segunda etapa. Esta na implementação do OpenCv é realizada em paralelo.

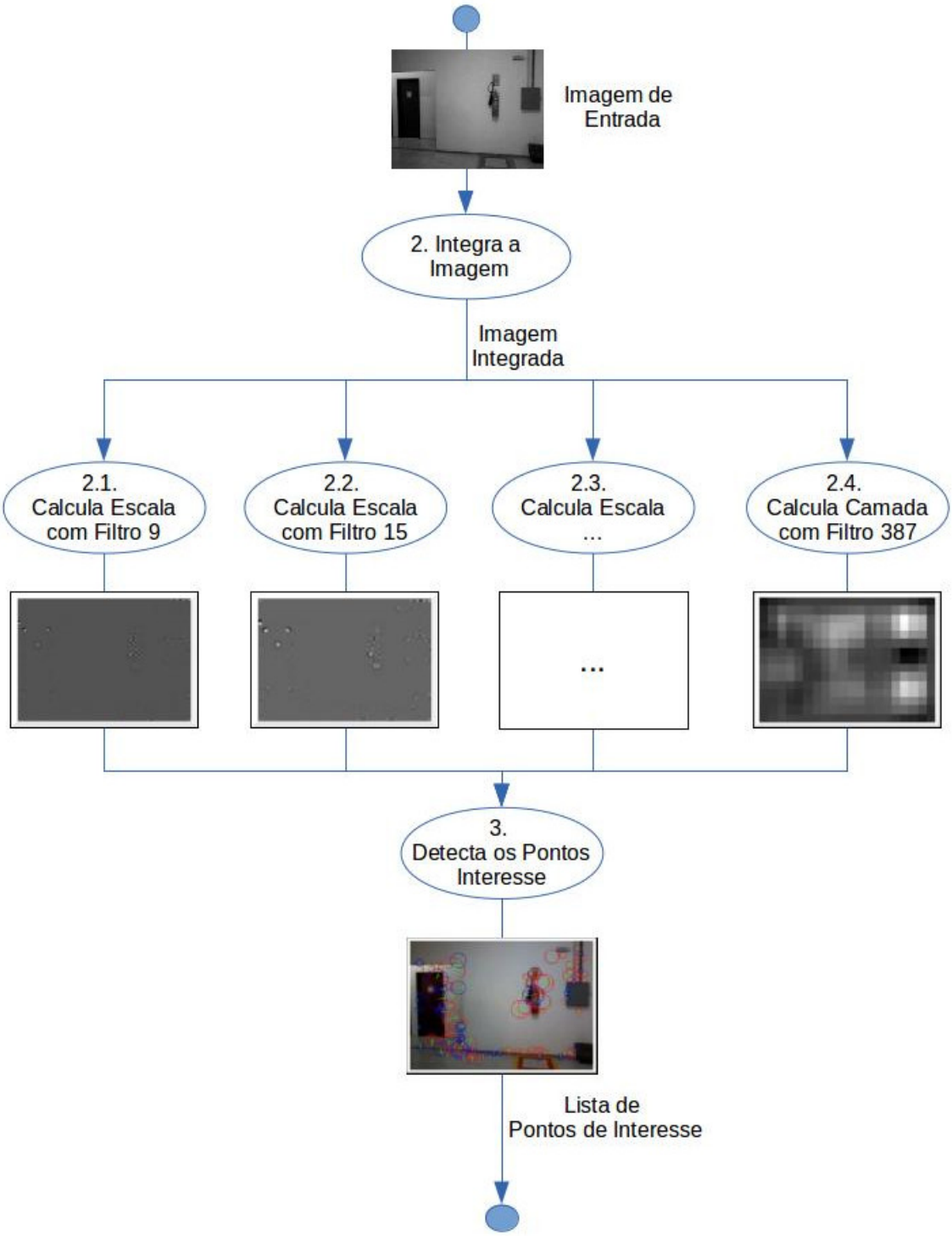


Figura 9: Diagrama do Algoritmo SURF baseado na implementação do OpenCv versão 2.4.9. Figura do autor.

A figura 10 mostra alguns resultados do algoritmo SURF, o qual detectou vários pontos sobre os cantos dos objetos, como pode-se vê-los nos cantos das portas, sofá e monitores, também detectou sobre objetos pequenos. Contudo existe o problema de ele retornar para um mesmo canto vários pontos em diferentes escalas, como pode ser

observado em vários pontos. Esta propriedade não é desejada para este problema, já que isso implica em um aumento na ambiguidade das correspondências, além de aumentar a base de dados desnecessariamente.

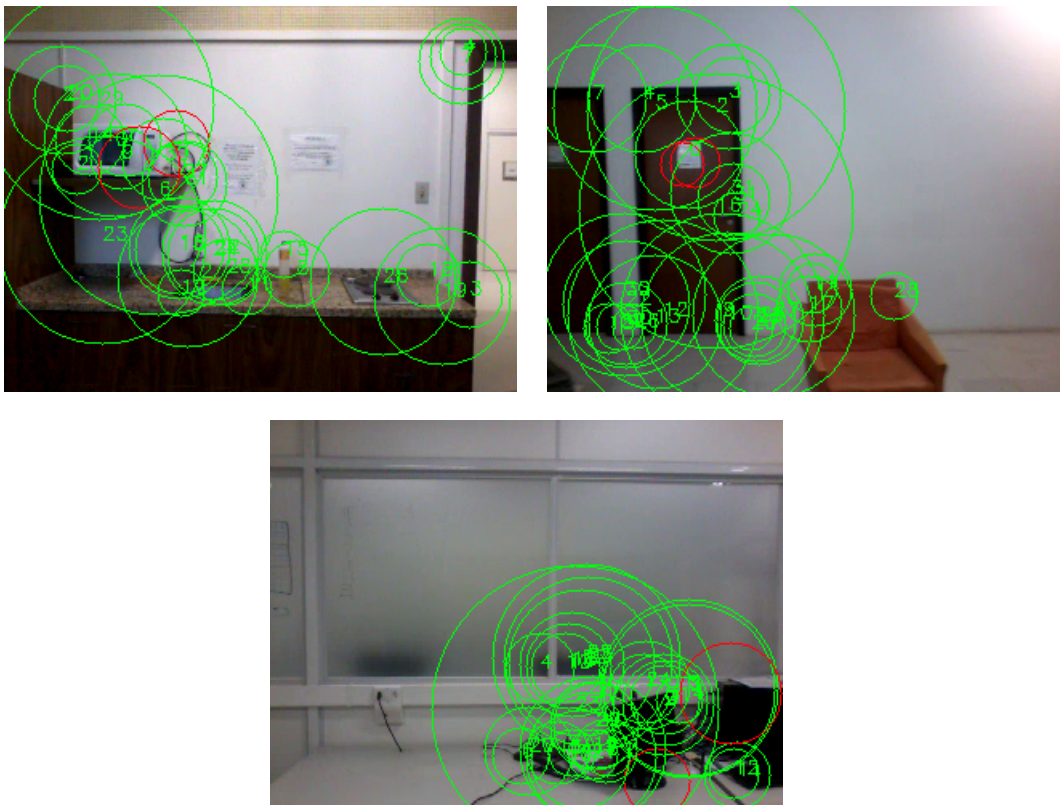

Figura 10: As figuras mostram alguns resultados do algoritmo SURF. Figura do autor.

## 2.4 Detector FAST

Este algoritmo foi elaborado por Edward Rosten e Tom Drummond na universidade de Cambridge na Inglaterra em 2006(12), e baseado na ideia do algoritmo SUSAN(13), o qual utiliza todos os pixeis de um circulo para verificar se ponto central corresponde a um canto ou não, enquanto o FAST usa somente as bordas do circulo, como representado na fig 11. Portanto ele tem um custo computacional menor. Além que ele percorre a imagem apenas uma vez, diferentemente de algoritmo como SIFT e SURF. Entretanto o tamanho do ponto detectado no FAST será sempre o mesmo, portanto não considera diferentes escalas.

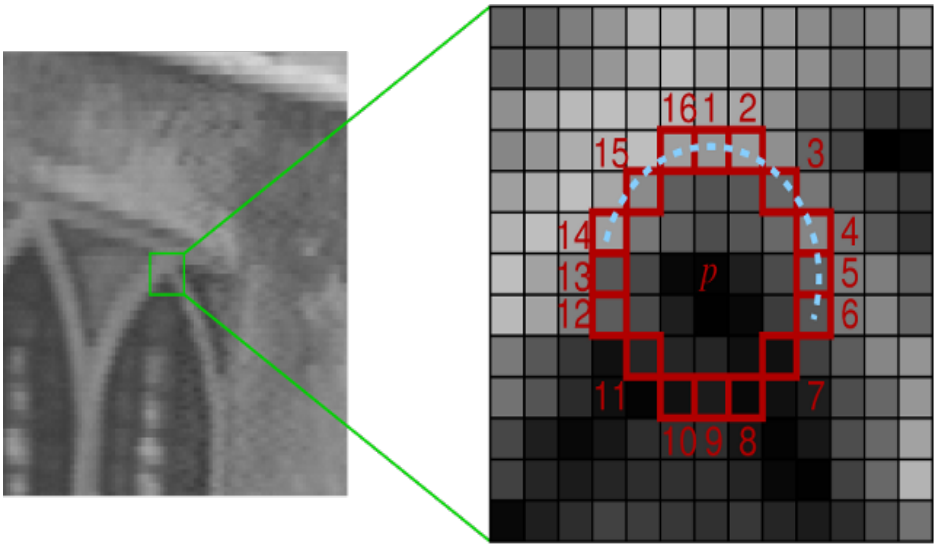


Figura 11: A figura a direita mostra a utilização das bordas do círculo de raio igual 3 para classificar o ponto P. Figura retirada do trabalho(12).

O algoritmo tem alguns parâmetros de entrada, como o raio do circulo, que pela implementação do OpenCv pode ser igual a 1 e corresponde a um circulo de 8 pixeis, 2 com 12 pixeis ou 3 com 16 pixeis. Por padrão utiliza-se 16 pixeis, o mesmo utilizado neste trabalho. Também existem 2 threshold como parâmetro para classificar se um ponto da borda é mais claro, mais escuro ou de mesma intensidade que o ponto central. E após a classificação dos 16 pixeis, ele determina se o ponto central é um canto ou não.

Também idêntico aos outros detectores de pontos, também existe o problema de encontrar muitos pontos de interesse na mesma região, o qual é resolvido da mesma maneira que os outros, utilizando a supressão de não máximos, ou seja, o ponto se tornará apenas ponto de interesse, se a força dele for maior que a de seus vizinhos.

## 2.5 Descritor SIFT

David Lowe seguiu um caminho completamente diferente para a elaboração do descritor do SIFT, já que antes de sua publicação, muitos trabalhos utilizavam um histograma de cores para comparar duas imagens. Então ele propôs a utilização de histograma de orientação como representado na figura 12 para servir como descritor.

Este descritor é formado por uma matriz de histogramas de orientação com dimensão de 4 por 4. Cada um destes histogramas é composto por 8 espaços, os quais representam as 8 possíveis orientações de um ponto.

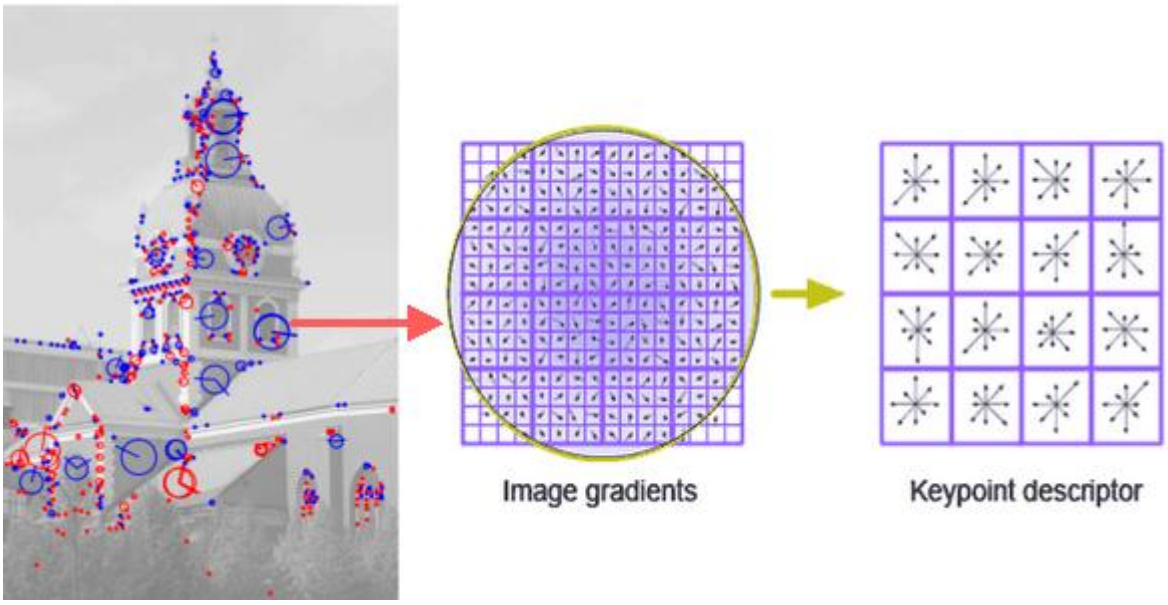


Figura 12: A figura a esquerda mostra uma imagem com os pontos detectados através do SIFT, logo em seguida é escolhido um dos pontos e mostrado a matriz 16x16 pontos com sua respectiva orientação e por final o descritor formado pelo 4x4 histogramas de orientação. Figura retirada do trabalho(5).

O calculo da orientação é realizado pela equação 2.8 e leva em conta também sua magnitude, a qual é calculada pela equação 2.9. Lembrando que $L(x, y)$ é a função que retorna o valor do ponto na coordenada (x,y) da imagem. Também é importante levar em consideração, que o algoritmo para a extração dos descritores tem como entrada a piramide gaussiana da imagem. Portanto para cada ponto de interesse é utilizado a imagem da oitava e da escala corresponde. Por final, para prover invariância a rotação, este é rotacionado pelo ângulo do ponto de interesse.

$$\theta(x, y) = \text{arctg}\left(L(x, y+1) - L(x, y-1), L(x+1, y) - L(x-1, y)\right) \tag{2.8}$$

$$m(x, y) = \sqrt{\left(L(x+1, y) - L(x-1, y)\right)^2 + \left(L(x, y+1) - L(x, y-1)\right)^2} \tag{2.9}$$

Importante frisar, que diferentemente do histograma de cores, o qual as cores com frequências mais altas terá um peso maior nos descritores, enquanto neste descritor as bordas terão um maior peso, pois elas conterão uma magnitude maior do que regiões com pouca mudança, como pode observar na fig. 13

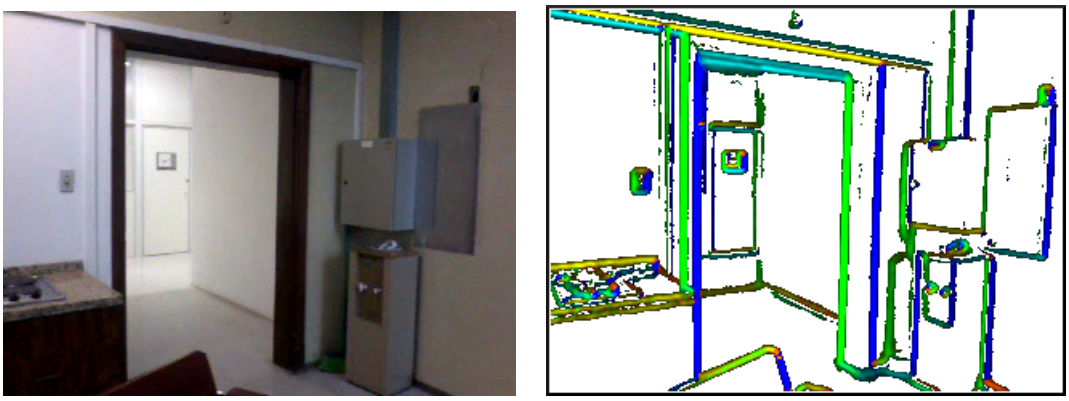

Figura 13: A figura a esquerda mostra a imagem original e a direita a orientação de cada ponto, sendo que a cor azul indica 0º, Amarelo 90º, Verde 180º e ciano 270º. Figura do autor.

A orientação dos pontos no decorrer dos quadros de um vídeo, apesar das mudanças de iluminação e variações dos pixeis devido ao ruído, mantêm-se bastante estável principalmente nos pontos com maior magnitude, pois o ruído deve ser muito alto para conseguir trocar sua orientação. Portanto esse descritor foi um grande avanço para o processamento de imagens, pois apresentou um método robusto invariante a rotação, pouco afetado pela variação de iluminação e pelo ruído, além de corresponder muito bem em diferentes escalas.

# 3 Implementação

## 3.1 Criação da Base de Dados

A elaboração da base de dados é realizada através de filmagens panorâmicas de 360º do ambiente usando uma câmera simples com resolução de 360x240 pontos. Para cada vídeo é executado o algoritmo mostrado na figura 14, o qual é composto por 3 partes principais. Primeiramente são detectadas as características de cada frame, utilizando um dos algoritmos de detecção citado neste trabalho, os quais são HARRIS, FAST, SIFT, SURF, MSER e o algoritmo de detecção proposto.

Os algoritmos HARRIS, FAST, SIFT e SURF utilizam o conceito de pontos de interesse amplamente utilizado em processamento de imagens. Enquanto o novo algoritmo proposto e o MSER utilizam as regiões estáveis da imagem, as quais correspondem a regiões sem bordas e normalmente composto por uma única cor. No passo seguinte, aplica-se um método para descrever todas as características, o que possibilita comparar e quantificar a similaridade de uma característica com outra, permitindo a busca na base de dados.

Por final, tem-se os descritores de todas as características de um quadro do vídeo, porém muitos deles já estão na base dados, pois já foram captados nos quadros anteriores. Então, como estes pontos repetidos aumentam o custo da pesquisa na base, sem um aumento expressivo da qualidade de classificação, é necessário detectar se algum ponto já existe na base, antes de inserí-lo.

Independente de qual método usado para detectar ou para descrever, o algoritmo de comparação é sempre o mesmo. Este é composto por uma busca linear do descritor da característica detectada com todos os descritores da base de dados de um ambiente utilizando distância euclidiana. E por final retorna o ponto com a menor distância. Ou seja, o algoritmo tem uma complexidade de $O(n * m)$, sendo n o número de descritores do quadro e m o número de descritores da base de dados. Este certamente não é o método de busca mais eficiente, mas este aspecto está fora do escopo deste trabalho.

Embora existem diversos descritores foi apenas utilizado o descritor SIFT de 128 elementos, pois apresenta uma maior unicidade das características, já que contém um número maior de elementos.

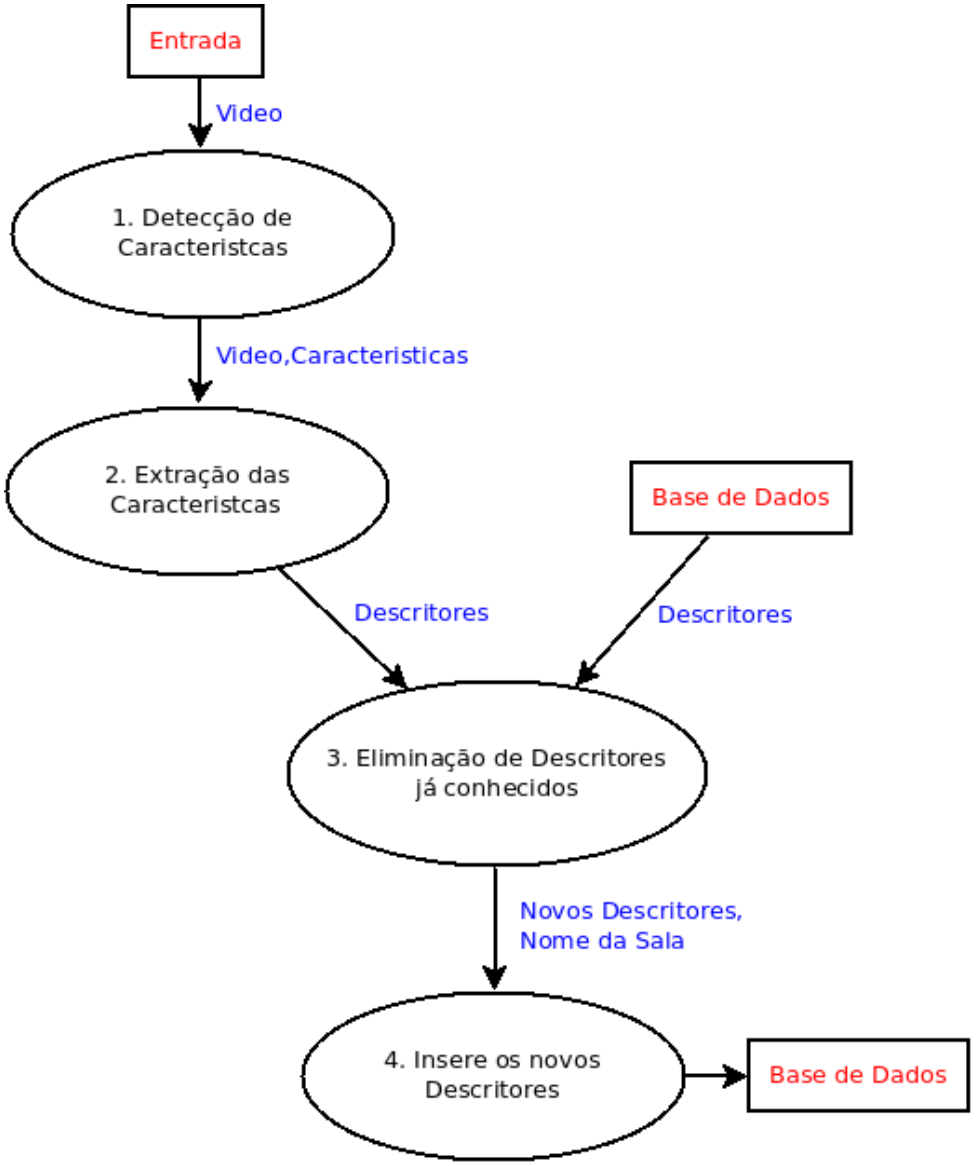


Figura 14: Algoritmo para elaboração da Base de Dados. Figura do autor.

## 3.2 Classificação do Video

A classificação recebe uma base de dados com os descritores das salas e um vídeo para classificar. Então ele prossegue quase da mesma maneira para criar uma base de dados. Com a diferença de que no último passo, após corresponder os pontos detectados em cada quadro do vídeo com os pontos na base de dados, o algoritmo incrementará o contador da sala para cada ponto correspondido, onde o ele foi encontrado. Ao final do vídeo a sala, que contiver o maior contador será elegida como o local onde o robô está.

Embora existem diversos caminhos para melhorar este resultado, como por exemplo trabalhar melhor a base de dados construída e eliminar ou diminuir o peso das características que são comuns a mais de um ambiente, ou até mesmo utilizar a informação dos ângulo dos

pontos juntamente com cadeias de Markov para eliminar possíveis falsas correspondências. Entretanto esta pesquisa tem maior foco na qualidade dos algoritmos de detecção das características e descritores associados para uso em localização por aparência visual.

## 3.3 Detector Proposto

Este novo método proposto para detecção de pontos de interesse leva em conta um detalhe notado sobre a posição de alguns pontos detectados através do método SURF, o qual detecta muitos pontos no vetor resultante, como mostrado na figura 15. Esses vários pontos no mesmo local ocorrem devido aos diversos filtros em diferentes escalas, ou seja, com o aumento do tamanho do filtro, desloca o ponto de interesse para o centro do retângulo. Então foi elaborado um algoritmo, o qual tem como objetivo detectar um único ponto para cada vetor resultante além de interligá-los de tal modo para prover ao algoritmo a propriedade de ser invariante à escala.

Também o novo método teve influência do algoritmo *Maximally Stable Extremal Regions*(MSER) proposto por Matas et al em 2002(14), que tem como objetivo detectar áreas com pouca textura, ou seja, áreas na imagem com pouca diferença de luminosidade e poucas bordas. Essa propriedade é conveniente a ambientes internos, pois este tipo de ambiente são formados por paredes, portas, chão e objetos com uma única cor, os quais tem pouca textura.

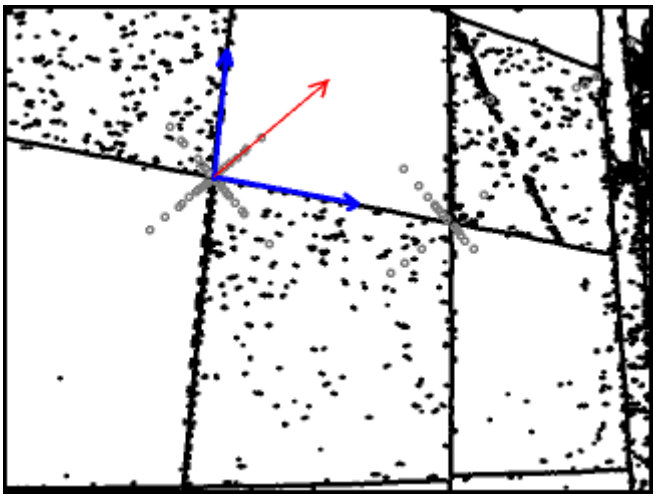

Figura 15: Os círculos cinzas representam a posição dos pontos de interesse detectado através do algoritmo SURF sobre as bordas da imagem obtida através do método Sobel. Figura do autor.

O algoritmo prosposto neste trabalho consiste em 5 partes, a primeira parte detecta as bordas através do algoritmo Canny(15), o qual utiliza um threshold adaptativo para

amenizar a mudança de iluminação. O resultado da detecção da borda pode ser visto na figura 16.

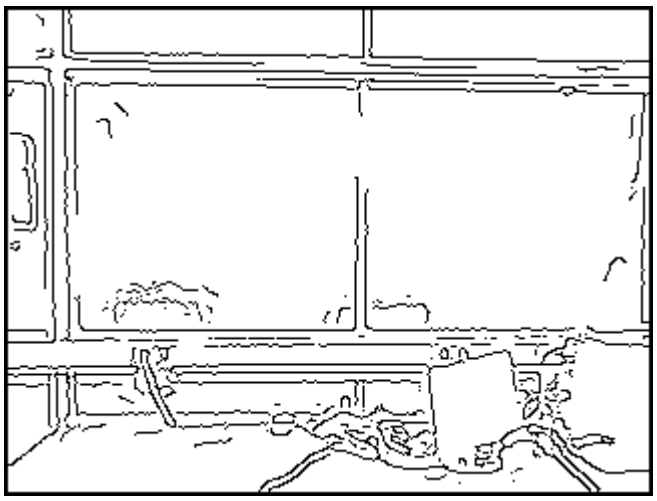

Figura 16: Bordas detectadas através do algoritmo Canny. Figura do autor.

Logo após, usa-se uma dilatação sobre a imagem binarizada para dilatar as bordas, assim obter os seus contornos e também para unir as retas, que são muito próximas. O resultado deste passo pode ser observado na figura 17.

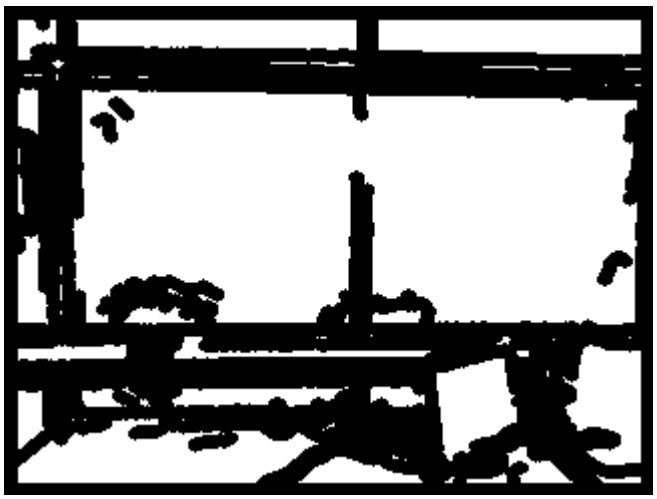

Figura 17: Bordas detectadas são dilatadas e assim unindo as bordas próximas e marcando o contorno de cada borda. Figura do autor.

O próximo passo é percorrer toda a imagem e quando encontrar uma borda, ou seja, um pixel preto com um pixel branco executa um algoritmo recursivo, que percorre toda a extensão da borda e para cada ponto é colocado em um vetor, o qual formará um contorno. Ao final do algoritmo é retornado um conjunto de contornos.

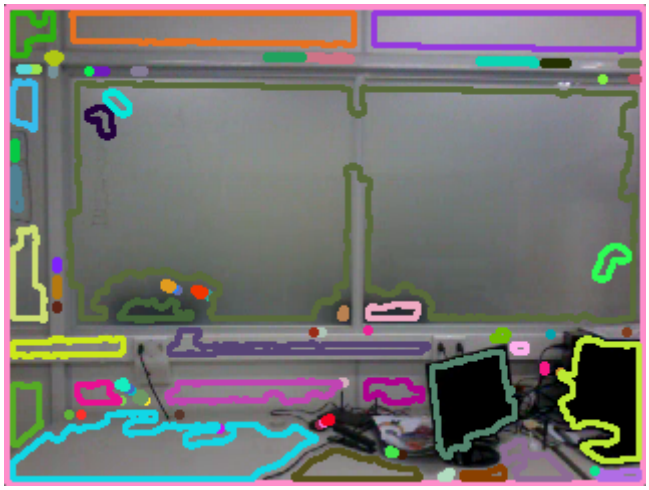

Figura 18: Figura mostra contornos detectados pelo algoritmo findContours da biblioteca OpenCv. Figura do autor.

Cada contorno contém muitos pontos, como pode ser visto na figura 18 , então para simplificar e formar um polígono com uma quantidade de pontos menor, é utilizado o método matemático Douglas-Peucker, o qual consiste em formar uma reta entre o primeiro e último ponto, e calcula a distância de todos os pontos até a reta formada, como mostrado na figura 19. Se o ponto mais distante da reta for maior que um threshold, então a reta é dividida em dois segmentos. Após a divisão é realizado o mesmo procedimento para os dois segmentos criados até que todos pontos estejam a uma distância menor que o threshold para reta correspondente.

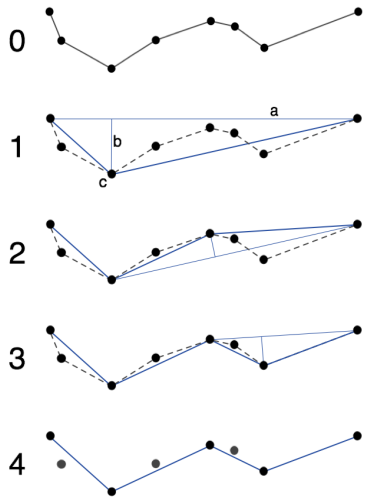


Figura 19: Mostra os passos do método Douglas-Peucker, o qual inicia no passo 0 com 8 pontos. No passo 1, a reta A liga os dois pontos extremos e B marca a distância do ponto mais distante da reta A. E ao final, no passo 4, o polígono é simplificado de 8 pontos para apenas 4. Figura retirada do site(16).

Portanto após a aplicação do método Douglas-Peucker, obtém-se vários polígonos como demostrado na imagem 20

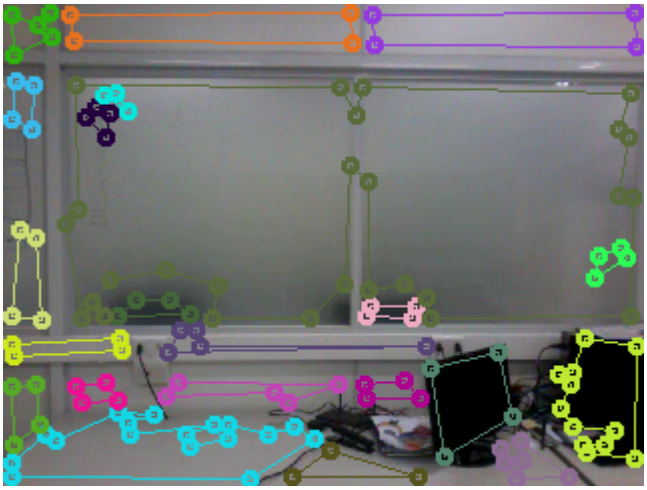

Figura 20: A figura mostra os polígonos com seus respectivos cantos e representam cada contorno detectado na figura 18. Figura do autor.

Por final calcula-se os 2 pontos mais extremos de cada polígono para formar um circulo, que contém todos os seus pontos. Este circulo será utilizado como ponto de interesse, o qual o centro será a posição e o raio o tamanho. Também é desenhado cada polígono em outra imagem. Isso para permitir, que dado um ponto da imagem saber em qual polígono ele corresponde. A partir desta informações pode-se calcular quantos pontos fazem parte da reta, além da cor média de cada região.

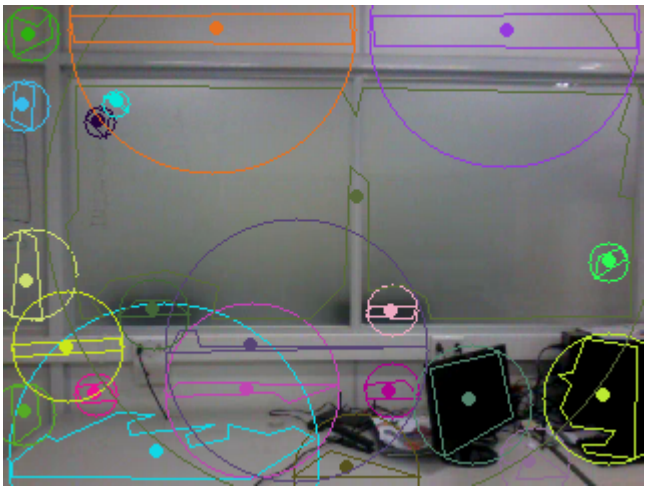

Figura 21: A figura mostra os círculos, que contém todos os pontos de cada regiões. Figura do autor.

## 3.4 Resultados

O método de avaliação consiste em elaborar uma base de dados com um vídeo de 360º filmado no centro de um ambiente, aqui no caso, foi escolhido o laboratório VRI.

Em um horário diferente é filmado um segundo vídeo de 360º também no centro do ambiente. Este indicará o quanto a iluminação e a mudança do ambiente influência nos resultados.

Também são filmados mais 4 vídeos de 360º nos cantos da sala como mostrado na figura 22. Estes indicarão o quanto a mudança da posição da câmera influência nos resultados. Por final foram realizados alguns vídeos fora do ambiente para poder comparar a porcentagem de correspondências dentro e fora do ambiente da base de dados.

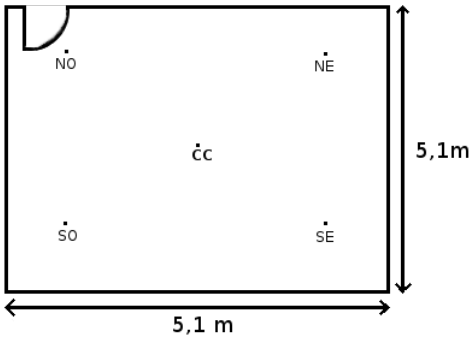


Figura 22: Planta baixa do laboratório VRI com a marcação de onde foi realizados os vídeos. Figura do autor.

A seguir há 6 tabelas, cada uma delas mostra o resultado dos testes utilizando os algoritmos HARRIS, FAST, SIFT, SURF, MSER e detector proposto respectivamente. Por exemplo, na primeira linha da tabela 1, mostra o nome do vídeo utilizado para o teste, o qual é vri-base, o mesmo utilizado para elaborar a base de dados. Logo após o número 32.582, que significa a quantidade total de pontos detectados em todo o vídeo. Em seguida o número de pontos correspondidos acompanhado pela porcentagem entre pontos correspondidos por pontos detectados. Logo após a largura com 2.465 descritores e seguido por uma porcentagem de 100%. Esta última porcentagem com 100% significa que foram encontrados, ao menos uma vez, todos os 2.465 descritores armazenados na base de dados.

A largura é um dado relevante, pois um vídeo pode ter vários pontos correspondidos entretanto e sempre do mesmo descritor. Portanto este descritor é muito genérico e não deve ser levando em conta. Também pode-se seguir no sentido contrário, um vídeo pode ter poucos pontos correspondidos mas com uma ampla largura, ou seja, vários descritores diferentes foram correspondidos, sendo que este caso ocorre na tabela 6.

Foram testados 9 vídeos, sendo que o vri-base foi o vídeo utilizado para elaborar a

base de dados. Já os 5 vídeos com prefixo vri-XX foram feitos dentro do laboratório VRI e o XX indica a posição da câmera na gravação como mostrado na figura 22. Enquanto o saguão, a cozinha e o hospital são lugares fora do VRI, sendo o saguão e a cozinha locais do mesmo edifício e o hospital, um local fora da universidade.

Tabela 1: Resultados com o detector HARRIS

| Video | Pontos Detectados | Pontos Correspondidos | | Largura | |
|---|---|---|---|---|---|
| vri-base | 32.582 | 32.582 | 100% | 2465 | 100% |
| | | | | | |
| vri-cc | 110.807 | 21.696 | 19,6% | 658 | 26,7% |
| vri-no | 47.537 | 4.599 | 9,7% | 270 | 11,0% |
| vri-ne | 59.491 | 6.291 | 10,6% | 441 | 17,9% |
| vri-so | 66.204 | 7.417 | 11,2% | 387 | 15,7% |
| vri-se | 47.555 | 6.769 | 14,2% | 254 | 10,3% |
| | | | | | |
| saguão | 9.469 | 2.225 | 21,8% | 201 | 8,2% |
| cozinha | 20.124 | 2.208 | 7,5% | 185 | 7,5% |
| hospital | 49.064 | 9.989 | 14,6% | 372 | 15,1% |

Tabela 2: Resultados com o detector FAST

| Video | Pontos Detectados | Pontos Correspondidos | | Largura | |
|---|---|---|---|---|---|
| vri-base | 43.926 | 43.926 | 100% | 3.287 | 100% |
| | | | | | |
| vri-cc | 182.952 | 15.232 | 8,3% | 512 | 15,6% |
| vri-no | 76.133 | 1.315 | 1,7% | 66 | 2,0% |
| vri-ne | 92.419 | 920 | 1,0% | 109 | 3,3% |
| vri-so | 107.361 | 1.617 | 1,5% | 115 | 3,5% |
| vri-se | 94.520 | 1.384 | 1,5% | 104 | 3,2% |
| | | | | | |
| saguão | 13.773 | 761 | 5,5% | 58 | 1,8% |
| cozinha | 25.436 | 520 | 2,0% | 40 | 1,2% |
| hospital | 84.079 | 2.209 | 2,6% | 91 | 2,7% |

Tabela 3: Resultados com o detector SIFT

| Video | Pontos Detectados | Pontos Correspondidos | | Largura | |
|---|---|---|---|---|---|
| vri-base | 26.956 | 26.956 | 100% | 2.742 | 100% |
| | | | | | |
| vri-cc | 122.232 | 14.609 | 12,0% | 591 | 21,6% |
| vri-no | 48.464 | 2.295 | 4,7% | 158 | 5,8% |
| vri-ne | 57.267 | 4.682 | 8,2% | 301 | 11,0% |
| vri-so | 67.772 | 4.769 | 7,0% | 300 | 10,9% |
| vri-se | 61.537 | 4.338 | 7,1% | 241 | 8,8% |
| | | | | | |
| saguão | 9.076 | 982 | 9,2% | 128 | 4,6% |
| cozinha | 15.861 | 1.233 | 7,7% | 142 | 5,1% |
| hospital | 53.668 | 3.302 | 6,1% | 260 | 9,5% |

Tabela 4: Resultados com o detector SIFT considerando apenas pontos de interesse com tamanho entre 10% a 60% do tamanho da diagonal da imagem

| Video | Pontos Detectados | Pontos Correspondidos | | Largura | |
|---|---|---|---|---|---|
| vri-base | 2516 | 2516 | 100,00% | 393 | 100,00% |
| | | | | | |
| vri-cc | 7937 | 239 | 3,01% | 32 | 8,14% |
| vri-no | 2821 | 3 | 0,11% | 2 | 0,51% |
| vri-ne | 3286 | 31 | 0,94% | 7 | 1,78% |
| vri-so | 3475 | 37 | 1,06% | 6 | 1,53% |
| vri-se | 3196 | 57 | 1,78% | 11 | 2,80% |
| | | | | | |
| saguão | 1119 | 28 | 2,50% | 5 | 1,27% |
| cozinha | 1628 | 5 | 0,31% | 4 | 1,02% |
| hospital | 3910 | 6 | 0,15% | 3 | 0,76% |

Tabela 5: Resultados com o detector SURF

| Video | Pontos Detectados | Pontos Correspondidos | | Largura | |
|---|---|---|---|---|---|
| vri-base | 50.018 | 50.018 | 100% | 4.120 | 100% |
| | | | | | |
| vri-cc | 225.747 | 31.053 | 13,8% | 1.113 | 27,0% |
| vri-no | 82.506 | 1.232 | 1,5% | 113 | 2,7% |
| vri-ne | 110.207 | 3.053 | 2,8% | 260 | 6,3% |
| vri-so | 119.409 | 3.998 | 3,4% | 260 | 6,3% |
| vri-se | 123.062 | 1.865 | 1,5% | 155 | 3,8% |
| | | | | | |
| saguão | 16.186 | 362 | 2,2% | 61 | 1,5% |
| cozinha | 36.476 | 1.357 | 3,7% | 54 | 1,3% |
| hospital | 102.850 | 1.362 | 1,3% | 75 | 1,8% |

Tabela 6: Resultados com o detector SURF considerando apenas pontos de interesse com tamanho entre 10% a 60% do tamanho da diagonal da imagem

| Video | Pontos Detectados | Pontos Correspondidos | | Largura | |
|---|---|---|---|---|---|
| vri-base | 50.018 | 50.018 | 100% | 4.120 | 100% |
| | | | | | |
| vri-cc | 149.441 | 24.894 | 16,66% | 1042 | 25,29% |
| vri-no | 52.748 | 936 | 1,77% | 87 | 2,11% |
| vri-ne | 69.897 | 2.190 | 3,13% | 178 | 4,32% |
| vri-so | 76.308 | 2740 | 3,59% | 225 | 5,46% |
| vri-se | 78271 | 1055 | 1,35% | 114 | 2,77% |
| | | | | | |
| saguão | 16.186 | 362 | 2,24% | 61 | 1,48% |
| cozinha | 36.476 | 1.357 | 3,72% | 54 | 1,31% |
| hospital | 102.850 | 1.362 | 1,32% | 75 | 1,82% |

Tabela 7: Resultados com o detector MSER

| Video | Pontos Detectados | Pontos Correspondidos | | Largura | |
|---|---|---|---|---|---|
| vri-base | 15890 | 15890 | 100,00% | 1605 | 100,00% |
| | | | | | |
| vri-cc | 60101 | 18636 | 31,0% | 489 | 30,4% |
| vri-no | 23984 | 5398 | 22,5% | 138 | 8,6% |
| vri-ne | 33375 | 3780 | 11,3% | 166 | 10,3% |
| vri-so | 34528 | 5198 | 15,0% | 203 | 12,6% |
| vri-se | 30325 | 3530 | 11,6% | 135 | 8,4% |
| | | | | | |
| saguão | 6957 | 677 | 9,7% | 52 | 3,2% |
| cozinha | 9854 | 383 | 3,8% | 33 | 2,1% |
| hospital | 40642 | 374 | 0,9% | 46 | 2,9% |

Tabela 8: Resultados com o detector proposto

| Video | Pontos Detectados | Pontos Correspondidos | | Largura | |
|---|---|---|---|---|---|
| vri-base | 7065 | 7065 | 100,0% | 665 | 100,0% |
| | | | | | |
| vri-cc | 17651 | 8229 | 46,6% | 294 | 44,2% |
| vri-no | 7537 | 2416 | 32,0% | 103 | 15,4% |
| vri-ne | 9107 | 2269 | 24,9% | 110 | 16,5% |
| vri-so | 9272 | 3236 | 34,9% | 138 | 20,7% |
| vri-se | 9889 | 1992 | 20,1% | 99 | 14,8% |
| | | | | | |
| saguão | 1721 | 133 | 7,7% | 21 | 3,2% |
| cozinha | 9854 | 116 | 3,2% | 23 | 3,5% |
| hospital | 3561 | 564 | 5,6% | 42 | 6,3% |

## 3.5 Discussão dos Resultados

A porcentagem dos pontos correspondidos nos vídeos realizados no laboratório VRI pelos algoritmos avaliados não foi muito elevado. Entretanto o que deve-se levar em conta é a diferença entre as porcentagens dos vídeos realizados dentro do VRI com vídeos realizados fora do VRI, pois isso indica se o algoritmo classificará corretamente ou não.

Os resultados obtidos pelos algoritmos de pontos de detecção tiveram uma boa resposta para o vídeo, vri-cc, o qual foi realizado na mesma posição que o vídeo base. Para este vídeo todos os 4 algoritmos de pontos de detecção desde HARRIS ao SURF

tiveram a porcentagem de correspondência maior que os vídeos realizados fora do VRI. Isto indica que o vri-cc tem maior semelhança ao vri-base, que os outros vídeos. Entretanto para os vídeos dentro do VRI filmados nos cantos da sala muitas vezes tiveram suas porcentagem de correspondência abaixo dos vídeos do saguão, cozinha e hospital. Ou seja, para o algoritmo o saguão, por exemplo, tem mais similaridade ao vri-base do que os vídeos realizados no VRI.

Já o algoritmo MSER, o qual tem finalidade de encontrar regiões estáveis, apresentou resultados melhores, já que a porcentagem de pontos correspondido nos 5 vídeos vri-XX foi maior do que a porcentagem de pontos nos 3 vídeos fora do ambiente. Além que a base de dados do MSER ficou menor que os algoritmos de ponto de interesse, essa vantagem resulta em uma busca pela base de dados mais rápida.

Por final, o algoritmo de detecção proposto obteve resultado melhores que o algoritmo MSER, já que teve uma maior correspondência para os 5 vídeos do ambiente e continuou com uma baixa porcentagem para vídeos fora do ambiente. Além de diminuir ainda mais o tamanho da base de dados.

# Conclusão

Os resultados dos algoritmos, que buscam por áreas estáveis como o MSER e o novo algoritmo proposto, apresentaram resultado melhores que os pontos de interesse. Já que, nos testes realizados, estes dois algoritmos apresentaram bastante diferença de porcentagem entre os vídeos realizados dentro do laboratório VRI dos vídeos realizados fora, logo tem uma probabilidade maior de classificar corretamente os vídeos.

Este resultado melhor para algoritmos de áreas estáveis ocorre devido aos ambientes internos frequentemente terem muitas paredes, portas, janelas, armários composto por uma única cor e portanto as imagens resultante tem pouca textura. Logo estes algoritmos detectam pontos melhores distribuídos pela imagem e normalmente pontos maiores, que os pontos detectados pelo algoritmos de pontos de interesse. Assim as áreas estáveis apresentam uma boa unicidade, já que houve um aumento na diferença das porcentagens de correspondência e também uma boa repetibilidade das características para mudança de perspectiva.

Enquanto o novo detector proposto teve resultados melhores ao MSER, além de retornar mais informações que outros algoritmos testados. Pois enquanto todos os algoritmos descritos neste trabalho retornam apenas a coordenada do ponto, sua força, tamanho, ângulo e oitava, o detector proposto retorna um conjunto de polígonos, o qual cada polígono tem seus subpontos e as retas, que formam a borda do polígono. Além de permitir, que a partir de uma coordenada da imagem, seja possível saber de qual poligono ele faz parte. Todas essas informações podem ser usadas para melhorar a unicidade dos descritores e portanto indica que o algoritmo proposto é promissor sendo interessante a pesquisa de aperfeiçoamento e validação do mesmo.

# Referências

1 Ulrich, Iwan; Nourbakhsh, Illah. Appearance-based place recognition for topological localization. *IEEE International Conference on Robotics and Automation*, p. 1023–1029, Abril 2000. Citado na página 7.

2 Chao, Zhou; Yucheng, Wei; Tieniu, Tan. Mobile robot self-localization based on global visual appearance features. *IEEE International Conference on Robotics & Automation*, p. 1271–1276, Setembro 2003. Citado na página 7.

3 ENGELSON, S. A. Using image signatures for place recognition. *IEEE International Conference on Robotics & Automation*, p. 941–951, 1998. Citado na página 7.

4 LOWE, D. Object recognition from local scale-invariant features. *Proc. of the International Conference on Computer Vision*, 1999. Citado na página 7.

5 LOWE, D. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, p. 91–110, 2004. Citado 5 vezes nas páginas 7, 11, 12, 13 e 20.

6 Harris, Chris ; Stephen, Mike. *A Combined Corner and Edge Detector*. 1988. Citado 2 vezes nas páginas 7 e 8.

7 Ke, Y.; Sukthankar, R. Pca-sift: A more distinctive representation for local image descriptors. *Proc. Conf. Computer Vision and Pattern*, p. 511–517, 2004. Citado na página 7.

8 Hashem Tamimi et al. Localization of mobile robots with omnidirectional vision using particle filter and iterative sift. *International Journal of Computer Vision*, p. 91–110, setembro 2006. Citado na página 7.

9 Frontoni, Emanuele ; Zingaretti, Primo. Visual feature group matching for autonomous robot localization. *14th International Conference on Image Analysis and Processing*, 2007. Citado na página 7.

10 Sobel, Irwin; Feldman, Gary. A 3x3 isotropic gradient operator for image processing. 1968. Citado na página 8.

11 BAY, T. T. H.; GOOL, L. V. Surf: Speeded up robust features. *9th European Conference on Computer Vision*, 2006. Citado 2 vezes nas páginas 14 e 15.

12 Rosten, Edward; Drummond, Tom. Machine learning for high speed corner detection. *9th European Conference on Computer Vision*, 2006. Citado 2 vezes nas páginas 18 e 19.

13 Smith, S. M.; Brady, J. M. *SUSAN — A New Approach to Low Level Image Processing*. 1995. Citado na página 18.

14 Matas, J. et al. Robust wide baseline stereo from maximally stable extremal regions. 2002. Citado na página 25.

15 CANNY, J. F. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis an Machine Intelligence*, PAMI-8, n. 6, p. 679–714, nov. 1986. Citado na página 25.

16 WIKIPEDIA. *Douglas-Peucker Algorithmus*. 2014. Disponível em: <http://de.wikipedia.org/wiki/Douglas-Peucker-Algorithmus>. Citado na página 27.