Odometria Visual

Paulo Lilles Jorge Drews Junior
Universidade Federal de Minas Gerais
Av. Antônio Carlos, 6627 - Pampulha - Belo Horizonte - MG
paulol@dcc.ufmg.br

Resumo

O presente artigo apresenta um método de odometria visual baseado em algoritmos descritores de caracteristica, no caso SIFT e SURF, tal método utiliza imagens calibradas de uma câmera presa a um robô em movimento e apartir disso reconhece a pose do robô, tal pose permite a localicazação do robô com o passar do tempo, sendo uma forma de odometria. Os resultados obtidos mostraram a determinação da odometria em um robô para inspeção de linhas de alta tensão subterraneas, percorrendo tubulações.

1. Introdução

A odometria convencional é um dos métodos mais utilizados para estimar a posição de um robô. Sabe-se bem que ela proporciona uma boa precisão em movimentos curtos e permite taxas de amostragem muito altas. A idéia fundamental da odometria é a integração de informação incremental do movimento ao longo do tempo, o qual envolve uma inevitável acumulação de erros. A acumulação de erros de orientação causa grandes erros na estimação da posição, os quais vão aumentando proporcionalmente com a distância percorrida pelo robô.

Esses erros são ainda maiores quando a odometria é fornecida por dados de encoder preso as rodas do robô devido a derrapagem da roda, que acontece normalmente, em maior ou menor quantidade, dependendo do local que o robô está andando e da característica das rodas. Outras fontes comuns de erros de odometria são quando o robô percorre subidas e descidas e quando ocorre movimentos de rotação.

Assim, a odometria visual surge como uma alternativa a odometria convencional, através da qual é possível estimar a posição do robô mesmo quando acontece derrapagem ou rotação, visto que os dados fornecidos pela visão não são sensíveis a esse tipo de problema.

Porém, a odometria visual é dependente do ambiente que o robô está inserido. Como a fonte sensora, utiliza-se as imagens advindas de câmeras, assim o ambiente precisa ter iluminação e *features* visuais de forma que o robô possa determinar como está se locomovendo, bem como o ambiente precisa ser estático ou com uma dinâmica muito lenta em relação ao robô.

O processo de odometria visual, pode ser divido em três grandes passos:

- **Detecção de Características na Imagem** - Neste passo são determinadas características distintivas que podem ser buscadas em diferentes imagens, como por exemplo, quinas(*corners*) , bolhas (*blobs*) ou cumes(*ridge*);
- **Correlação de Características** - As diferentes características determinadas em imagens, precisam ser correlacionadas, ou seja, é preciso determinar quais características são as mesmas em imagens distintas;
- **Estimação do Movimento do Robô** - Através dos pares de pontos correlacionados em imagens distintas, tornar possível determinar o movimento da câmera, tanto de translação quanto de rotação, considerando conhecida a conversão do sistema de coordenada da câmera para o do robô, pode-se então inferir o movimento do robô.

2. Revisão Bibliográfica

Diversos trabalhos tem sido propostos para realizar odometria visual, com as mais diversas abordagens para os mais diferentes tipos de ambientes. Considerando a divisão da odometria visual em três passos, foi feita uma revisão bibliográfica para cada um dos três passos, bem como de trabalhos que fundem os três passos, afim de realizarem a odometria visual.

Para a determinação de características o método mais comum é detector de Harris. O detector de extremidades de Harris-Stephens [10], também conhecido como detector de Plessey, têm como intenção inicial a localização de quinas em uma imagem através da matriz Hessiana, para posterior correlação. Embora seja um método largamente utilizado,

ele é muito suscetível ao gradiente da imagem, tornando assim difícil a tarefa de determinar um limiar ótimo para determinação de *corners*.

Outro método para determinação de características é detector de congruência de fase [16]. Tal detector utiliza medidas de energia para localizar bordas e quinas. Tal medida, são invariantes ao contraste da imagem. Além disso, tal detector permite ainda detectar bordas e quinas simultaneamente, possibilitando o uso cooperativo de ambas informações.

Existem alguns métodos que são capazes de, além de determinar características, também criarem descritores que têm informações relevantes sobre tais, de tal forma que seja possível identificar elas, somente com a informação destes descritores. Os mais importantes são o o SIFT [17], o GLOH [19], o SURF [2] e o LESH [25]. Tais descritores costumam ser invariantes a diversas transformações, como rotação, translação, escala, além de possuírem robustez a ruído e iluminação.

A detecção de características com o SIFT [17] é um método de detecção de características invariante à diferentes rotações e escalas de uma imagem, além de funcionar bem para consideráveis distorções na imagem, adição de ruídos e mudança de iluminação. Além do método ser capaz de determinar um número muito grande de pontos, o que facilita o processo posterior de correlação. O SIFT tem um desempenho superior a diversos algoritmos, como mostra [19], embora sua maior limitação seja o custo computacional elevado.

O GLOH (*Gradient Location and Orientation Histogram*) [19] é uma versão mais robusta e especializada do SIFT. Este considera mais regiões espaciais para o histograma. Enquanto o SIFT gera oito gradientes de orientação, o GLOH gera 16, definindo assim um histograma de 272 regiões. O tamanho do descritor é reduzido com PCA (*Principal Component Analysis*), uma técnica usada para reduzir dados multi-dimensionais a fim de analisá-los. De modo geral o GLOH apresenta uma maior eficiência na detecção de características em relação ao SIFT, porém seus resultados não são muito superiores. Contudo, em cenas com textura ou quando as retas da imagem não são bem definidas, o SIFT tem um melhor desempenho. O custo computacional do GLOH também é elevado, sendo maior que do SIFT.

O SURF (*Speeded Up Robust Features*) [2] é um descritor de características, inspirado parcialmente no SIFT, visando um menor custo computacional com boa performance. Os autores propõem 3 versões do descritor do ponto, porém o método de extração é o mesmo para todos. Ele se baseia no uso de integral de imagens a do uso do *Fast-Hessian*, através de uma aproximação do kernel gaussiano de segunda ordem, para determinar características. Para determinação do descritor é utilizado a soma das respostas do *2D Haar wavelet* em diferentes orientações. O versão padrão do SURF cria um descritor de 64 posições, sendo muito eficiente porém menos preciso. Uma variante deste, ainda menos custosa computacionalmente não corrige a orientação do ponto. Existe uma terceira versão que tem um descritor de 128 posições considerando assim os diferentes sinais do *2D Haar wavelet*, esta sendo muito precisa.

O LESH (*Local Energy based Shape Histogram*) [25] é um robusto descritor de imagens, embora essas possam ser ”pequenas”, o que difere dos outros descritores que caracterizam, a princípio, regiões de interesse. Ele pode ser usado para adquirir eficientemente a descrição de formas. O LESH, então, constrói o modelo de energia local da imagem utilizando a congruência de fase [16]. Com esse modelo, ele determina o descritor das características, acumulando a energia local ao longo de várias orientações. Criando assim, histogramas locais das diferentes partes da imagem, no caso 16 partes, pela concatenação destes gera-se um descritor de 128 posições, este tendo alta tolerância a ruído e iluminação, além de ser invariante a escala. Pouco se conhece sobre a performance temporal do LESH, embora acredite-se ser muito custoso computacionalmente.

Para realizar a correlação de características, existem diversos métodos. Considerando os métodos de determinação de características que constroem descritores, a correlação é, comumente, feita através da distância entre os vetores dos descritores, utilizando distância euclidiana ou de Mahalanobis[2]. Assim sendo o tamanho do descritor tem interferência direta no custo do algoritmo.

Existem ainda algumas considerações sobre o processo de correlação utilizando descritores de pontos. O SIFT além de considerar a menor distância entre descritores, ainda faz uma consideração com a segunda menor distância, esta devendo ser maior que uma razão pré-definida. O SURF para fazer *matching* considera o sinal do laplaciano, ou seja, o *trace* da matriz hessiana. O SURF usa, ainda, uma abordagem semelhante ao SIFT, considerando a distância euclidiana e a razão com a segunda menor distância. Para fazer a correlação, o LESH transforma os descritores num espaço de similaridade de pose (PSFS - *pose similarity feature space*) esse espaço é computado através da similaridade entre os descritores do LESH. Para tal, é utilizada uma versão modificada do método da divergência K-L, tal método é numericamente estável e robusto a ruído. Este método fornecendo uma medida de dissimilaridade entre os histogramas, permitindo encontrar os histogramas correlacionados.

Existem ainda outras formas de realizar correlação, diferente dos descritores de características. Estas podem ser baseadas em intensidade, como por exemplo o *Correlation Score* [9]. Outra forma de realizar correlação é através de técnicas baseadas em análise de

textura para a determinação de correspondência entre pontos são muito utilizadas, diversos operadores de textura são citados na bibliografia, como *Co-ocurrence Matrix* [24], *Energy Filter* [23], *Local Binary Pattern* [13] [22], *Contrast Features* [21], *Symmetric Covariances* [12].

Outras medidas de largamente utilizada para realizar correlação é através do fluxo ótico. Nessa abordagem, não se busca pontos em duas imagens consecutivas e, posteriormente, se faz a correlação entre pontos. Nessa abordagem se faz-se um *tracking* das características baseado no modelo do fluxo ótico, o critério de similaridade para localização de pontos correlatos é baseado no método de Newton para minimização da soma das diferenças quadráticas(SSD) dentro de uma janela de busca ao redor da posição da característica(ponto de interesse) buscada na próxima imagem, assumindo um modelo de movimento translacional entre imagens subseqüentes. Este método é bastante eficiente quando se tem movimentos lentos entre os frames, possibilitando que o ponto esteja na janela de busca. O método é eficiente, porém ele necessita de um *refresh* custoso computacionalmente para determinar os pontos a serem buscados, se esse processo for realizado com alta freqüência, o método pode ter seu desempenho degradado.

A última parte de um sistema de odometria visual é a determinação do movimento, a partir dos pares de pontos correlacionados. Existem diversas abordagem, para a inferência do movimento. Tal abordagem depende do tipo de visão a ser utilizada, monocular ou estéreo [20], de forma a se determinar a posição 3D do robô. As abordagens costumam utilizar triangulação entre duas câmeras, ou entre pontos correlacionados em 3 imagens diferentes. A matriz essencial também é comumente utilizada de forma a se determinar a posição 3D [6], através das matrizes R e T, que representar rotação e translação, respectivamente. A abordagem utilizando visão estéreo além de ser maior precisa, ainda oferece a facilidade de não necessitar verificar pontos em 3 imagens distintas [20].

O trabalho de [18] apresenta uma solução de odometria visual, aplicado a robôs com limitação de processamento e energia, tal informação é fundida com a odometria das rodas e informação inercial fornecida por uma IMU. O sistema utiliza um par de câmeras estéreo, no qual tenta buscar características em imagens através do detector de Harris, sendo buscadas com a maior precisão possível, sem a necessidade de determinar muitas características, nem muito rapidamente, assim limitando o movimento do robô. Para determinar o deslocamento a partir das características, é utilizado um metódo iterativo, que se não convergir, é mantida a informação apenas da odometria das rodas.O sistema é utilizado, principalmente, em terrenos arenosos, onde existe muita derrapagem, em terrenos com desníveis muito grande e ao transpor pequenos obstáculos.

O trabalho proposto por [20] estima a odometria visual através de uma cabeça estéreo ou de uma câmera de vídeo se movendo junto ao robô. Tal sistema tem performance em tempo-real, tal sistema utiliza o detector de Harris também. Para correlação é utilizado um método semelhante ao KLT, sem precisão de subpixel. O método RANSAC é utilizado visando uma medida robusta do movimento, tal determinado por triangulação. Os resultados mostram o método proposto pode ser utilizado em diversos ambientes, tendo resultados muito bons frente a INS e GPS. Além disso o trabalho apresenta o RANSAC como uma solução não-linear e multi-modal, visto que o erro da odometria visual costuma ter essas características, provando assim que o filtro de Kalman não é suficiente para resolver o problema.

3. Metodologia

A metodologia proposta neste trabalho utilizará os seguintes passos para determinar a odometria visual:

- Calibração da Câmera e Correção das Distorções do Sistema Óptico
- Determinação de Pontos de Interesse em Imagens Utilizando Algoritmos Descritores de Features (SIFT e SURF)
- Correlação Entre Descritores de Features
- Remoção Robusta de Outliers, utilizando RANSAC/LMedS
- Determinação do Movimento Com Visão Monocular Utilizando Matriz Essencial

Um diagrama para melhor ilustrar a metodologia proposta é apresentado na figura 1. Ela mostra os passos necessários para a obtenção da pose da câmera em relação a um referencial inercial, assim conhecida a conversão da pose de câmera é possível determinar a pose do robô.

Na seções a seguir serão detalhados cada passo da metodologia.

3.1. Calibração da Câmera e Correção das Distorções do Sistema Óptico

Para a realização da calibração da câmera, utilizou-se implementação do método de Zhang [27] através da biblioteca OpenCV [15], para calibração da câmera, ou seja, para a determinação dos parâmetros intrínsecos além das constantes k de distorção da câmera.

Antes de executar a calibração propriamente dita, é necessário realizar alguns passos. Primeiro, o método requer que a câmara observe um padrão planar com diferentes (pelo menos três) orientações/posições. Assim é necessário se obter uma seqüencia de imagens desse

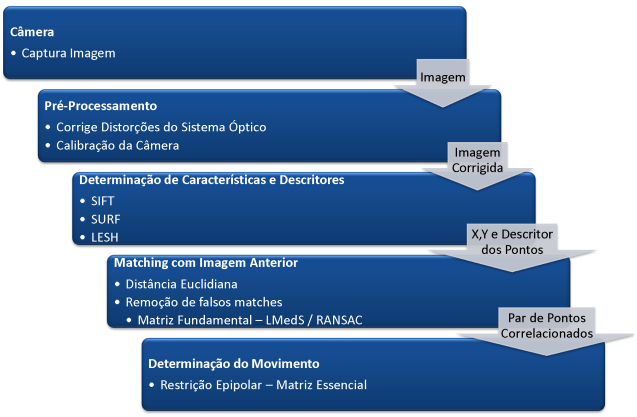

Figura 1. *Overview* do sistema proposto.

padrão. Nesse trabalho foi utilizado o formato quadriculado, semelhante ao utilizado em tabuleiros de xadrez, com 13×9 quadrados. A figura 2 mostra o padrão utilizado.

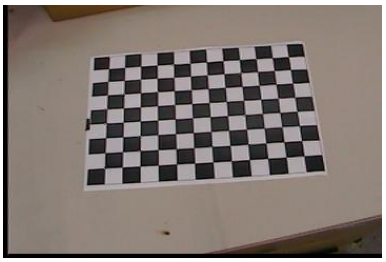

Figura 2. Padrão de Calibração Utilizado.

Depois de adquiridas as imagens do padrão de calibração, se detectam intersecções entre os quadrados. Inicialmente é utilizada morfologia matemática, através de um método de dilatação seguido por uma limiarização para eliminação de ruídos. Após esse processo são detectados contornos e, posteriormente, polígonos, utilizando o método Douglas-Peucker [7], a intersecção dos vértices desses polígonos são os pontos de interesse para a calibração.

Através do método para determinação da intersecção dos pontos se consegue uma precisão de meio pixel, porém para a determinação precisa da localização, utilizou-se de uma técnica de sub-pixel, que permite uma precisão de 0.1 pixel na localização, através de um método iterativo que tenta minimizar o erro de posição do ponto através de interpolação com a vizinhança do mesmo, utilizando o gradiente [4] [1].

Com a localização precisa dos *corners* do padrão quadriculado, é então realizado o processo de calibração, propriamente dito. O método proposto consiste em determinar uma solução analítica, que permite obter uma aproximação inicial dos parâmetros intrínsecos e extrínsecos da câmara, seguida de um refinamento não-linear baseado no critério de *máxima verossimilhança*. Ao final, é incluído no modelo a distorção radial da lente, dadas as soluções analítica e não-linear e então determinados tais coeficientes.

Com os coeficientes de distorção radial, é determinada a nova localização do ponto, através de equação 1 truncada em seu termo quadrático, onde o ponto $m = (x, y)$ representa as coordenadas distorcidas, o ponto $m^{'} = (x^{'}, y^{'})$ as coordenadas corrigidas, além de k_1 e k_2 serem os coeficientes de distorção adquiridos durante o processo de calibração.

$$\begin{aligned} x^{'} &= x + x \cdot [k_1 \cdot (x^2 + y^2) + k_2 \cdot (x^2 + y^2)^2] \\ y^{'} &= y + y \cdot [k_1 \cdot (x^2 + y^2) + k_2 \cdot (x^2 + y^2)^2] \end{aligned} \quad (1)$$

Com as coordenadas corrigidas de cada ponto, criou-se uma LUT, ou seja, uma tabela associativa computada offline, que define para cada pixel da imagem sua nova posição, gerando assim a imagem corrigida com baixo custo computacional.

3.2. Determinação de Pontos de Interesse em Imagens Utilizando Algoritmos Descritores de Features (SIFT e SURF)

Para a determinação de pontos de interesse nas imagens foram escolhidos os algoritmos descritores de *features*, tais algoritmos costumam ser invariantes a diversas características presentes em imagens, como iluminação, rotação, escala, translação, dentre outras. Tornando assim, tais algoritmos adequados a tarefa de determinação de pontos para a odometria visual, considerando que podemos ter robôs dinâmicas relativamente rápidas, tendo assim variações significativas entre cada frames.

O sistema permite utilizar o SURF e o SIFT, dependendo da aplicação dada ao sistema, cada um com suas vantagens e desvantagens. Assim, cada algoritmo será detalhado a seguir.

3.2.1. SIFT é um método robusto para extrair e descrever pontos-chaves de uma imagem [17]. O algoritmo é composto por 4 etapas:

1. **Detecção de extremos no espaço de escala:** Nessa etapa, é feita a busca em todas as escalas e localizações de imagens com diferença de filtros gaussianos (DOG,

do inglês *difference of gaussian*), visando identificar pontos de interesse invariáveis à escala e rotação.

2. **Localização de pontos-chaves:** Para cada localização em que foi detectado um extremo, um modelo detalhado é ajustado para determinar a localização exata e a escala. Pontos-chaves são selecionados baseando-se em medidas de estabilidade. Nessa etapa são definidos os melhores pontos para o sistema de mapeamento, por meio de medidas de gradiente.
3. **Definição de orientação:** A orientação é definida para cada ponto chave por meio dos gradientes locais da imagem. Toda operação, a partir de então, será feita com relação a dados da imagem transformados em relação à orientação e escala de cada ponto-chave. Desta maneira, obtém-se invariância a estas transformações.
4. **Descritor dos pontos-chaves:** O gradiente local de cada ponto-chave é medido, utilizando-se a vizinhança do ponto. Estas medidas são transformadas para uma representação que permite tolerância a níveis significativos de distorção e mudança de iluminação.

O algoritmo SIFT busca pontos que sejam invariantes a mudanças de escala da imagem, possibilitando a detecção de pontos em diferentes visões de um mesmo objeto, com altas taxas de repetibilidade, através de uma localização de pontos em uma espaço de escala. O primeiro estágio encontra extremos nesse espaço, localizando na função $D(x, y, \theta)$, ou seja, a função Diferença Gaussiana (DoG). Esta função pode ser computada pela diferença de duas imagens em uma dada escala separadas por um fator multiplicador k, como mostra a equação 2, onde $L(x, y, \sigma)$ é a imagem no espaço de escala, construída através da convolução da imagem $I(x, y)$ com o kernel gaussiano $G(x, y, \sigma)$. A figure 3 ilustra esta operação.

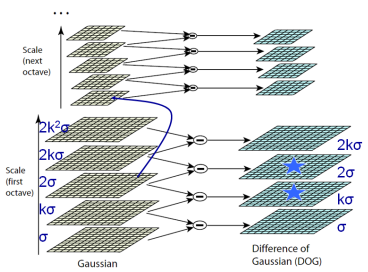

Figura 3. Espaço de Escala - Diferença Gaussiana(DoG)

São computados então extremos locais, ou seja, pontos que são máximos ou mínimos em uma vizinhança $3 \times 3 \times 3$ de tal forma que se consulta a escala anterior e posterior, como mostra figura 3 são buscados pontos de extremos nas imagens com uma estrela desenhada no DoG. Pontos que são extremos locais em uma função DoG são considerados pontos-chaves, ou *keypoints*. A geração de extremos, neste estágio, depende da freqüência de amostragem do espaço de escala k e do fator de suavização inicial σ_0. Os *keypoints* são, então, filtrados através da determinação da estabilidade [5].

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned} \quad (2)$$

Antes do descritor do ponto ser construído, é determinada a orientação dos *keypoints*, de forma que o descritor seja invariante a rotação. Essa orientação é calculada através de um histograma de gradientes locais. Para cada imagem amostrada em uma dada escala, o gradiente de magnitude $m(x, y)$ e a orientação $\theta(x, y)$ são computados usando diferença de pixels, como mostram as equações 3 e 4.

$$\begin{aligned} m(x, y) = (L(x+1, y) - L(x-1, y))^2 + \\ (L(x, y+1) - L(x, y-1))^2)^{1/2} \end{aligned} \quad (3)$$

$$\begin{aligned} \theta(x, y) = tan^{-1}((L(x, y+1) - L(x, y-1)) \\ /(L(x+1, y) - L(x-1, y))) \end{aligned} \quad (4)$$

Cada valor adicionado ao histograma tem um peso determinado pela magnitude do gradiente e pela escala em que o ponto se encontra. *Keypoints* adicionais são gerados, caso o ponto tenha múltiplas orientações dominantes,ou seja, mais de um pico no histograma com magnitude superior a 80% em relação ao valor máximo.

Após escolhidos os pontos-chave, invariantes a rotação e escala, um descritor para cada um desses pontos, baseados nos histogramas da região em torno do ponto, deve ser computado. São definidas então, regiões nxn em torno do ponto com kxk pixels, Lowe sugere que valores de n e k igual a quatro obtém melhores resultados [17], então constroem-se um histograma, semelhante ao utilizado em na determinação da orientação, porém com 8 direções apenas, que definirá o descritor.

O descritor é representado por um vetor, onde o valor de cada posição do vetor se refere a uma das direções dos histogramas. Com valor de n e k iguais a quatro, obtém-se um vetor com 128 posições. Para evitar efeitos de borda na qual o descritor abruptamente muda enquanto a amostra se desloca suavemente indo de um histograma para outro ou de uma orientação para outra, uma interpolação trilinear é utilizada para distribuir o valor de cada amostra de gradiente para sua posição no histograma adjacente.A figura 4 ilustra um descritor porém com $n = 2$ e $k = 4$.

Para uma maior tolerância a variação de iluminação, o descritor é normalizado. Após a normalização, todos os valores acima de um determinado limiar são ajustados para este

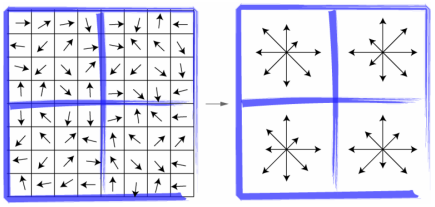

Figura 4. Descritor local do SIFT

limiar. Isto é feito para que direções com magnitude muito grande não dominem a representação do descritor. Lowe sugere utilizar um limiar de 0.2 [17], ocorrendo uma normalização, novamente.

3.2.2. SURF é outro método robusto e com baixo custo computacional para extrair e descrever pontos-chaves de uma imagem [2]. O algoritmo é composto por 3 etapas:

- Criação da Integral da Imagem
- Determinação de Pontos de Interesse através da *Fast-Hessian*
- Criação do Descritor de Cada Ponto-chave

O conceito de integral de imagem foi desenvolvido por [26], tal tipo de imagem permite uma implementação extremamente de mascaras de convolução. Tal representação, se utiliza do fato que características retangulares em imagens podem ser computadas muito rapidamente utilizando essa representação. A integral da imagem ($I_{\sum}$ pode ser calculada para um ponto $X = (x, y)$ na imagem $I(1..m, 1..n)$ através da equação 5, que representa a soma de todos os pixels localizados na parte superior esquerda do pixel.

$$I_{\sum}(X) = \sum_{i=1}^{i \leq x} \sum_{j=1}^{j \leq y} I(i, j) \tag{5}$$

Porém pode-se implementar eficientemente a integral da imagem considerando apenas a soma dos três valores anteriores, à esquerda, acima e na diagonal superior esquerda. Uma vez tendo calculada a integral de imagem, somente é necessária quatro operações de forma a determinar a soma de qualquer região quadrada na imagem, independente do tamanho da mesma. A figura 5 ilustra o processo de uso da integral da imagel, a área $S = A - B - C + D$.

Para a determinação dos pontos de interesse são necessárias duas etapas, primeiramente se determina a *Fast-Hessian* para cada escala diferente, e posteriormente se percorre as imagens resultados para se determinar qual os pontos de interesse que serão calculados seus descritores.

O processo de implementação da *Fast-Hessian* pode ser visto como uma simplificação do cálculo da matriz Hessiana, com a melhora de desempenho proporcionada pela

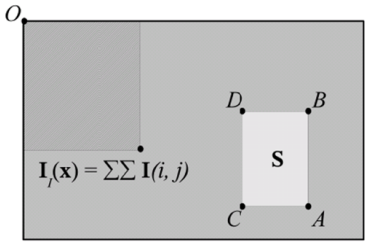

Figura 5. Forma de Calcular o Somatório dos Pixeis de uma Área Quadrada S a partir da Integral da Imagem

integral da imagem. Assim, como querermos determinar pontos invariantes a escala, devemos procura-los em diversas escalas, determinando em qual assim em qual das diversas escala o ponto realmente se encontra. Para análise de pontos no espaço de escala, *kernels* gaussianos são ótimos. Assim, para determinação da *Fast-Hessian* da imagem, utilizaremos *kernels* gaussianos de segunda ordem, tal discretizado e cortado, conforme mostra a figura 6. O exemplo mostra um kernel 9×9, representando $\sigma = 1.2$, ou seja, nossa menor escala(maior imagem). Para buscarmos pontos invariantes a escala, na pior das hipóteses, precisamos buscar até kernel do tamanho da imagem, ou seja $m \times n$, como a cada oitava temos o kernel duas vezes maiores, então termos $\log_2 min(m, n)$ oitavas no pior caso, cada oitava representa a imagem reduzida pela metade, ou seja, $I(1..\frac{m}{2}, 1..\frac{n}{2})$, porém no caso do SURF não reduzimos a imagem, e sim aumentamos o tamanho do Kernel, tendo o mesmo efeito, além de nos aproveitarmos da integral de imagem, que mesmo aumentando o kernel não aumentamos a complexidade assintótica. A figura 7 mostra o principio utilizado pelo SURF.

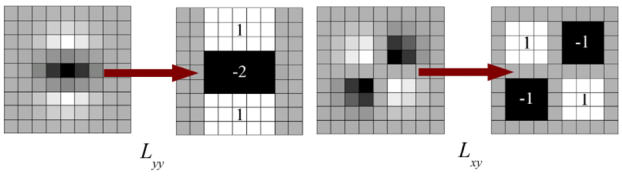

Figura 6. Aproximação de Kernel Gaussiano de Segunda Ordem, L_yy e L_xy

Com a *Fast-Hessian* determinada, é preciso agora determinar quais pontos são pontos relevantes para uma possível análise de correlação. Assim, são localizados pontos de máximos e mínimos em uma região $3 \times 3 \times 3$, sendo en-

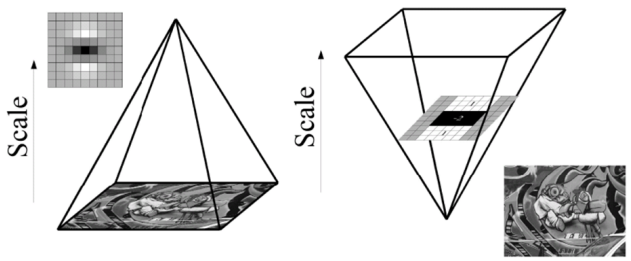

Figura 7. Amostragens em Diversas Escalas Utilizada pelo SURF, à Direita, Diferente da Abordagem Tradicional à esquerda

tão buscados na escala anterior, escala atual e escala posterior.A figura 8 mostra os pontos analisados para determinação de pontos de interesse.

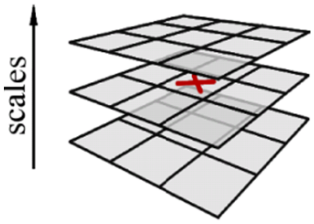

Figura 8. Determinação de Pontos de Interesse no SURF

Após os pontos de interesse serem determinados são computados os descritores para cada um deles. [2] propõe o uso de 3 variações do SURF, a primeira a versão padrão detalhada abaixo, a segunda com custo computacional menor e variante a rotação (USURF) porém muito util para movimentos lentos entre frames. Uma terceira versão do SURF só que com descritor do ponto maior, contendo mais informação, porém com custo maior para realização do *matching*, detalhado posteriormente.

Para tornar o SURF invariante a rotação, é calculada a resposta ao kernel *Haar-wavelet* nas direções x e y e de seus vizinhos em uma circunferência de raio $6s$, sendo s a escala do ponto. Após calculada a *wavelet* ele é "pesada" por uma gaussiana com $\sigma = 2.5s$. A orientação é estimada através do calculo da soma de todas as respostas através de uma janela deslizante de $\frac{\pi}{3}$.

O descritor propriamente dito é extraido através de alguns passos. Primeiramente é construída uma região quadrada centrada no ponto de interesse e orientada segunda orientação determinada no passo anterior, tal janela tendo tamanho $20s$. A janela é dividida em sub-regiões menores de tamanho 4×4, sendo então utilizada a resposta ao kernel *Haar-wavelet* nas direções x e y. Tais respostas são somadas para cada subregião, formando assim o primeiro conjunto de entradas do descritor. É também calculado o somatório dos valores absolutos, assim são formandos os descritores, considerando as 16 regiões de tamanho $4x4$, cada qual gerando 4 valores (somatório da resposta em x, y e dos valores absolutos em x e y), tem-se então um descritor com 64 posições.

A figure 9 ilustra a resposta gerada pelo uso do kernel *Haar-wavelet*, bem como o processo de formação do descritor. Uma versão alternativa do SURF, com 128 posições é proposta também, tal se diferencia do SURF padrão por determinar os somatórios para as respostas em x e y, diferenciando valores positivos dos negativos da coordenada contrária, ou seja, são somados as respostas em x e os valores absolutos das respostas em x para valores positivos de y e y no ponto, e vice-versa, gerando então 8 valores por região 4×4, gerando um descritor de 128 posições.

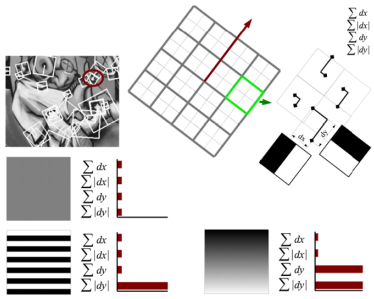

Figura 9. Formação do Descritor do SURF e Resposta ao kernel *Haar-wavelet*

3.3. Correlação Entre Descritores de Features

Para encontrarem-se correspondências entre duas imagens, conhecendo-se os descritores, é necessário localizar os melhores candidatos a serem seus equivalentes na outra imagem. Isto é feito procurando-se o vizinho mais próximo ou *nearest neighbor* do descritor do ponto entre todos os possíveis candidatos. Quando se procura classificar um ponto em um extenso banco de dados de descritores para vários objetos, a busca exaustiva do vizinho mais próximo pode ser demorada, assim Lowe propõe o uso de uma variação do *K-D Tree* [8], o BBF [3] para acelerar a busca [17].

Para determinar-se o vizinho mais próximo, é utilizada a mínima distância euclidiana entre os descritores. Porém, nem sempre o vizinho mais próximo é o ponto procurado, assim é necessária a utilização da distância euclidiana com

o segundo vizinho mais próximo, sendo excluídas correlações que tem a razão entre o vizinho mais próximo e o segundo maior que um determinado limiar, Lowe propõe 0.8, removendo assim 90% de falsos verdadeiros(*outliers*) e apenas 5% de correlações corretas [17].

Tal abordagem é utilizada em ambos os descritores utilizados, porém no SURF também é utilizado o sinal do laplaciano de forma a realizar correlação. Assim, para realizar correlação no SURF não se utilizou o BBF, e sim uma busca exastiva, visto que o descritor é menor e com uso do laplaciano tornando uma medida eficiente.

3.4. Remoção Robusta de Outliers, utilizando RANSAC/LMedS

Algumas vezes, pode acontecer do par definido pela correlação não se o par correto, assim, o sistema deve ser capaz de identificar e remover essas falsas correlações. Uma solução para o problema é através da geometria epipolar, que será explicada brevemente a seguir, visto que a única restrição geométrica entre dois pares de imagens não calibrados que existe é a geometria epipolar.

Considerando a geometria epipolar podemos calcular a matriz fundamental. Sendo, $\tilde{m} = [x, y, 1]$ um ponto sobre o plano imagem I e $\tilde{m}^{'} = [x^{'}, y^{'}, 1]$ um ponto sobre o plano imagem $I^{'}$, assim a equação 6 define tal matriz.

$$\tilde{m}^T F \tilde{m'} = 0 \tag{6}$$

Diversos métodos para estimação da matriz fundamental são encontrados na bibliografia, contudo o método mais conhecido é o algoritmo de 8 pontos [11], tal método, dado um conjunto com $n \geq 8$ correspondências, estima a matriz fundamental de forma linear, solucionando a equação 7.

$$\sum_{i=1}^{n} ||\tilde{m_i}^T F \tilde{m_i}^{'}||^2 \tag{7}$$

A estimação robusta da matriz fundamental é feita pesando o residual para cada ponto, o resíduo é mostrado pela equação 8, onde r o resíduo e i o número do par de pontos na lista de pontos correlacionados. Muitas funções peso diferentes foram propostas, cada uma sendo uma nova variação do método. Os resultados obtidos são muito bons na presença de *outliers*, mas são ruins quando os pontos não estão bem localizados. Dentre as funções mais utilizadas estão a RANSAC e a LMedS, tais utilizadas nesse trabalho.

O LMedS e o RANSAC são técnicas muito similares. Ambas as técnicas são baseadas na seleção randômica de um conjunto de pontos que são usados para a estimação linear da matriz fundamental. A diferença entre eles é a forma de determinar a melhor estimação da matriz fundamental. O LMeds calcula a distância entre os pontos e a linha epipolar, para cada estimação de F, para escolher a melhor matriz, o método minimiza a mediana dos valores. Enquanto o RANSAC calcula o número de inliers, ou verdadeiras correlações, para cada matriz F e escolhe a que maximiza esse número. Tendo eliminado os *outliers*, a matriz F é recalculada com o objetivo de obter melhorar a estimação.

$$r_i = \tilde{m_i}^T F \tilde{m_i}^{'} \tag{8}$$

3.5. Determinação do Movimento Com Visão Monocular Utilizando Matriz Essencial

Uma vez tendo computado a matriz fundamental F e com a matriz de parâmetros intrinsicos da câmera conhecidos K, pode-se determina a pose da câmera entre dois frames subsequentes, permitindo assim também a reconstrução dos pontos no espaço 3D.

A posição e orientação da câmera pode ser considerado como um par de matriz de rotação e um vetor de translação R, t. Assim, computar a pose do é equivalente a encontrar a matriz de rotação $R = R_2$ e o vetor de translação $t = t_2$ da câmera entre a figura I_1 e a figura I_2, considerando conhecida a posição inicial de I_1, com $R = I_{3\times3}$ e $t = 0$.

Para encontrar R e t, utiliza-se a matriz essencial E que é conhecida como sendo $E = K_1 F K_2^T$, porém em nosso caso temos a equação 9 visto que as duas vistas são da mesma câmera, então K é constante.

$$E = KFK^T \tag{9}$$

As propriedades da matriz essencial E asseguram que ela é diagonalizavél. Consequentemente, é possível realizar uma decomposição SVD na matriz E. Pode-se considerar U e V as matrizes tais que existe uma diagonal $D = diag(\lambda_1, \lambda_2, \lambda_3)$ tal que $\lambda_1 \geq \lambda_2 \geq \lambda_3$. No caso da matriz essencial temos que $\lambda_3 \approx 0$. Assim a equacao 10 define a relação.

$$E = UDV^T \tag{10}$$

Assim, podemos estipular o par (R, t) através de E. Existem quatro possibilidades para os pares, porém consideramos somente duas, visto que as outras duas possibilidades são inviáveis a principio. A equação 11 nos mostra como determinar o par e as duas possibilidades de R.

$$R^1 = U \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} V^T$$

$$R^2 = U \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} V^T$$

$$T = V \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} V^T$$

$$T = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix} \quad (11)$$

Assim temos agora o problema de determinar quais das duas matriz de rotação é a correta, R^1 ou R^2. Uma solução encontrada foi escolher a matriz que mais se aproximava da identidade, utilizando assim $||R - I||$. Essa solução se baseia no principio de que temos movimentos pequenos entre cada frame, assim a matriz deve se aproximar da identidade se estiver correta, ou seja, da rotação nula. Porém, algumas vezes nenhuma das duas matrizes de rotação calculas se aproximam da matriz essencial, nesse caso então a matriz de rotação está próxima da matriz identidade oposta, ou seja, $-I_{3\times3}$. Assim, mudamos R^1, R^2 e t de sinal e recalculamos os valores novamente para determinar qual a matriz de rotação correta.

Com as matrizes de translação e rotação conhecidas determinamos a pose da câmera e, conseqüentemente, a pose do robô a menos de um fator de escala, conhecido como universal, este pode ser determinado através de uma calibração da cena.

4. Resultados

O presente trabalho foi desenvolvido utilizando-se a linguagem de programação C++, orientada a objetos, juntamente com o Qt, um framework para desenvolvimento de aplicações C++. Para o desenvolvimento da interface gráfica utilizou-se a ferramenta *Qt Designer*, que permite a criação de interfaces gráficas em alto nível. A biblioteca OpenCV foi largamente utilizada devido a performance e facilidade de uso[15], também se utilizou de uma implementação do SIFT em C desenvolvida por [14] e da implementação do SURF desenvolvida por [2].

Foi realizados testes para validar o sistema em um robô para inspeção de linhas de alta-tensão subterrâneas, a figura 10 mostra um protótipo do robô que foi utilizado nos testes. Embora nesta figura não esteja acoplado ao robô a câmera de vídeo, foi utilizada uma câmera de vídeo Samsung SCD-363 no padrão NTSC, que permite uma taxa máxima de $29,97$ fps e tamanho da imagem de 720×480 pixels. Porém, foram utilizados frames de 320×240 pixels, para melhorar a resposta temporal do sistema. A figure 11 mostra a câmera utilizada.

Para a calibração da câmera, se utilizou um padrão plano quadriculado de 13×9 quadrados de 21cm, através da opencv levantou os parâmetros intrínsecos, incluindo os coeficientes de distorção radial da lente. A figura 12 mostra o resultado de retificação da imagem, tal processo ocorre em toda imagem capturada da câmera. Embora a câmera de vídeo utilizada apresente baixos coeficientes de distorção, a tabela 1 mostra os coeficientes da câmera.

Figura 10. Robô para Inspeção de Linhas de Alta-tensão Subterrâneas Utilizado nos Testes.

Figura 11. Câmera de Vídeo Utilizada Nos Testes

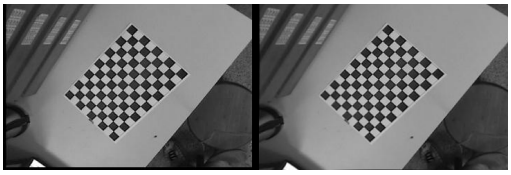

Figura 12. Imagem do Padrão Plano Antes e Após a Correção

Com o sistema calibrado, foi determinado o movimento do robô em uma tubulação, tal movimento predominantemente de translação visto que o robô pode apenas rotacionar em torno do tubo ou transladar nele, devido a restrições de construção do mesmo. Devido ao fato do robô não possuir qualquer outra fonte sensora para determinação de posição e orientação não foi possível estimar sua trajetória, porém sabe-se que ele percorreu 2.5m (tamanho da tubulação percorrida pelo robô). As figuras 13 e 14 mostram os resultados da odometria visual utilizando as 3 versões do SURF

Modelo da Câmera	f_1	f_2	u_0	v_0	k_1	k_2
Samsung SCD-363	458.7	418.2	195.4	102.4	-0.18	0.22

Tabela 1. Parâmetros de Calibração da Câmeras Utilizada

e o SIFT, primeiramente sem eliminação de *outliers* e com eliminação.

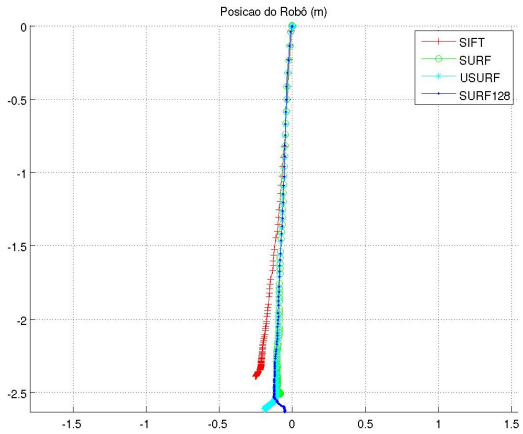

Figura 13. Resultado da Odometria Sem Remoção de *Outliers*

Outros resultados importantes estão nas figures 15, 16, 17 e 18. Tais figures representam, respectivamente, o número de características determinadas pelo SURF e SIFT(o número de característica é invariante ao uso dos 3 tipos de descritores do SURF), o número de correlações realizadas, de verdadeiras correlações realizadas (após a remoção de *outliers*) e o tempo gasto para executar todo processo para cada frame.

Os resultados mostram a superioridade temporal do SIFT, gastando muito mais tempo que todos os tipos de SURF, mesmo assim não tendo um resultado tão superior na determinação da posição. O SIFT também detectou bem mais pontos, assim a posição determinada pela estimação robusta foi bem melhor, chegando muito próximo ao ótimo. Já o SURF, apresentou os resultados esperados, sendo mais rápido que o SIFT porém detectando bem menor características (metade), conseqüentemente, menos correlações foram efetuadas, embora o resultado tenha sido bom, chegando bem próximo aos

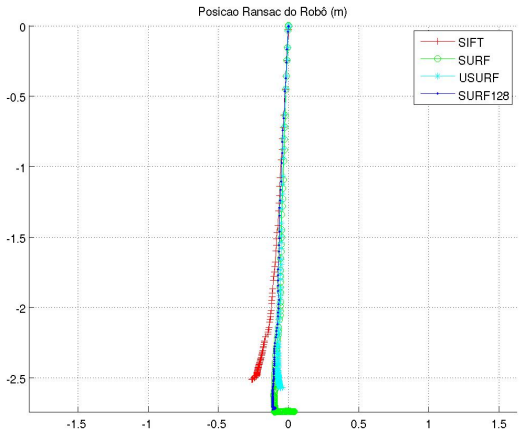

Figura 14. Resultado da Odometria Com Remoção de *Outliers* através do RANSAC

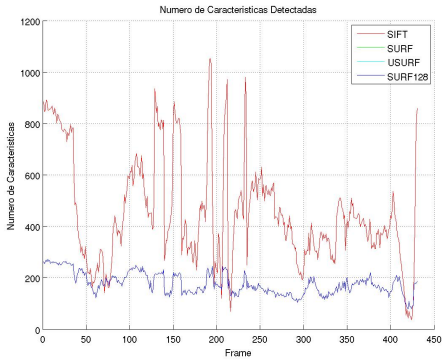

Figura 15. Numero de Características Detectadas em Cada Frame

2.5m.

5. Conclusão e Trabalhos Futuros

O artigo mostrou um sistema para realização de odometria através de um sistema de câmera monocular, para isso se utilizou-se de algoritmos descritores de características, permitindo assim determinas pontos na imagem que fossem descritivos e que pudessem ser encontrados em outras imagens. A correlação de descritores foi realizada, sem descartados *outliers*, ou falsos verdadeiros, utilizando técnica de estimação robusta da matriz fundamental RANSAC e LMedS.Para a identificação da pose da câmera e, conseqüentemente do robô, utilizou-se a matriz essencial, esti-

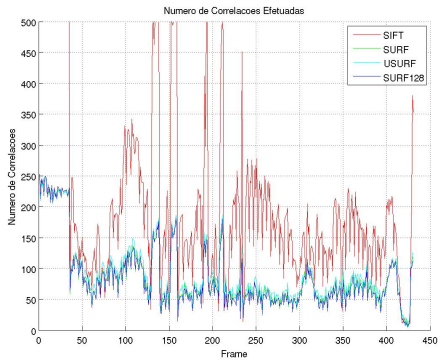

Figura 16. Numero de Correlações de Cada Frame

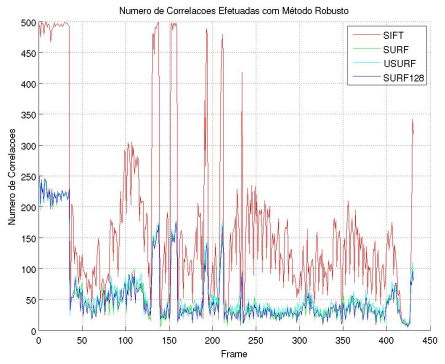

Figura 17. Numero de Correlações "Verdadeiras" de Cada Frame

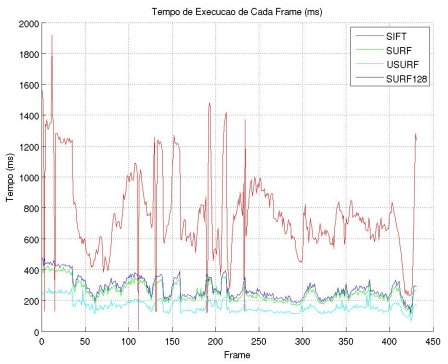

Figura 18. Tempo de Execução de Cada Frame

mada a partir da matriz fundamental.

Os resultados mostram que ambos os métodos tem uma boa resposta ao movimento do robô, muito embora o ambiente não tenha muitas características (quinas) e a iluminação seja não uniforme(artificial).

Como trabalhos futuros, pretende-se testar a odometria visual em outros ambientes e um estudo mais aprofundado da necessidade de adaptação do método proposto para funcionar nestes ambientes. Também a utilizada de estimadores robustos de posição e fusão sensorial com outras formas de sensores, permitindo assim uma estimação precisa da posição.

Referências

[1] C. Achard, E. Bigorgne, and J. Devars. A sub-pixel and multispectral corner detector. In *15th International Conference on Pattern Recognition*, volume 3, pages 959–962, 2000.

[2] H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. In *European Conference on Computer Vision*, pages 404–417, 2006.

[3] J. Beis and D. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Conference on Computer Vision and Pattern Recognition*, pages 1000–1006, Washington, DC, USA, 1997. IEEE Computer Society.

[4] J.-Y. Bouguet. Pyramidal implementation of the lucas kanade feature tracker. Technical report, Intel Corporation, Microprocessor Research Labs, 2000.

[5] M. Brown and D. Lowe. Invariant features from interest point groups. In *British Machine Vision Conference*, pages 656–665, Cardiff, Wales, 2002.

[6] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, June 2005.

[7] D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Canadian Cartographer*, 10:112–122, 1973.

[8] J. Friedman, J. Bentley, and R. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, 1977.

[9] A. Giachetti. Matching techniques to compute image motion. *Image and Vision Computing*, 18(3):247–260, February 2000.

[10] C. Harris and M. Stephens. A combined corner and edge detection. In *Fourth Alvey Vision Conference*, 1988.

[11] R. Hartley. In defense of the eight-point algorithm. *Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593, June 1997.

[12] D. Harwood, T. Ojala, M. Pietikäinen, S. Kelman, and L. Davis. Texture classification by center-symmetric autocorrelation, using kullback discrimination of distributions. *Pattern Recognition*, 16(1):1–10, 1995.

[13] D.-C. He and L. Wang. Textural filters based on the texture spectrum. *Pattern Recognition*, 24(12):1187–1195, 1991.

[14] R. Hess. Sift 1.1.1: A c implementation of scale invariant feature transform. http://web.engr.oregonstate.edu/ hess/index.html, 2007. [Acessado em 08 de Novembro de 2007.].

[15] Intel. The open computer vision library. http://www.intel.com/technology/computing/opencv/, 1999. [Acessado em 1 de Abril de 2007.].

[16] P. Kovesi. Phase congruency detects corners and edges. In *IAPR Digital Image Computing Techniques and Applications*, pages 309–318, 2003.

[17] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[18] M. Maimone, Y. Cheng, and L. Matthies. Two years of visual odometry on the mars exploration rovers: Field reports. *Journal of Field Robot*, 24(3):169–186, 2007.

[19] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis e Machine Intelligence*, 27(10):1615–1630, 2005.

[20] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1), January 2006.

[21] T. Ojala and M. Pietikäinen. Unsupervised texture segmentation using feature distributions. In *9th International Conference on Image Analysis and Processing*, pages 311–318, London, UK, 1997. Springer-Verlag.

[22] T. Ojala, M. Pietikäinen, and D. Harwood. A comparative study of texture measures with classification based on feature distributions". *Pattern Recognition*, 29(1):51–59, January 1996.

[23] T. Randen and J. Husoy. Texture segmentation using filters with optimized energy separation. *IEEE Transactions on Image Processing*, 8(4):571–582, April 1999.

[24] S. S. Aksoy and R. Haralick. Textural features for image database retrieval. In *IEEE Workshop on Content-Based Access of Image and Video Libraries*, pages 45–49, June 1998.

[25] M. S. Sarfraz and O. Hellwich. Head pose estimation in face recognition across pose scenarios. In *International conference on Computer Vision Theory and Applications*, pages 235–242, January 2008.

[26] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *IEEE Conference on Computer Vision and Pattern Recognition*, 1:511–518, 2001.

[27] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, November 2000.