ODOMETRIA VISUAL MONOCULAR PARA LOCALIZAÇÃO DE ROBÔ TERRESTRE

Renan Guedes Maidana*, Aurelio Tergolina Salton†, Alexandre de Morais Amory*

*Pontifícia Universidade Católica do Rio Grande do Sul
Faculdade de Informática
Porto Alegre, RS, Brasil

†Pontifícia Universidade Católica do Rio Grande do Sul
Faculdade de Engenharia
Porto Alegre, RS, Brasil

Emails: renan.maidana@acad.pucrs.br, aurelio.salton@pucrs.br, alexandre.amory@pucrs.br

Abstract— The problem of localization is intrinsic to Mobile Robotics, described as the estimation of an agent's (e.g. robot, car, etc) position. Various methods of position estimation have been developed, among which Visual Odometry consists of reconstructing the agent's trajectory through a sequence of images from one or more cameras. In this paper, we study the method of Monocular Visual Odometry, as well as its peculiarities and the subjects related to its implementation. We also implemented a Monocular Visual Odometry algorithm, used to estimate a robot's trajectory in an indoor environment with good results.

Keywords— Visual Odometry, Mobile Robotics, Localization, Computer Vision, Pose Estimation.

Resumo— O problema da localização é intrínseco à Robótica Móvel, definido como a estimação da postura de um agente (por exemplo, robô, carro, etc). Diversos métodos de estimação de postura foram desenvolvidos, dentre os quais o método da Odometria Visual consiste em reconstruir a trajetória de tal agente através de sequências de imagens de uma ou mais câmeras. Neste trabalho, foi estudado o método de Odometria Visual Monocular, assim como suas peculiaridades e os assuntos relacionados à sua implementação. Foi realizada também a implementação de um algoritmo de Odometria Visual Monocular, utilizado para estimar a trajetória de um robô terrestre em um ambiente interno, com resultados favoráveis.

Palavras-chave— Odometria Visual, Robótica Móvel, Localização, Visão Computacional, Estimação de postura.

1 Introdução

Na área da robótica móvel, existe uma grande demanda de pesquisa para sistemas robóticos autônomos, cujo funcionamento pleno requer pouca ou nenhuma interação com agentes externos. Dentro desta área, existe o problema da localização, definido como a determinação da postura de um robô no espaço através de sensores.

Uma das formas mais simples de obter-se a localização de um robô terrestre é utilizando medidas de odometria, que consiste na integração de estimações de postura ao longo do tempo. Com isto, é possível obter a postura atual do robô através de informações de seus movimentos anteriores. A técnica de Odometria Visual (do inglês, Visual Odometry, VO) consiste em inferir a postura do robô por meio de imagens de uma ou mais câmeras (Scaramuzza and Fraundorfer, 2011). Esta técnica permite a navegação de terrenos instáveis (por exemplo, arenosos) ou com desníveis, onde odometria com encoders (no caso de robôs terrestres) sofre com deslizes das rodas, por exemplo.

Este artigo tem por objetivo realizar um estudo da odometria visual, em especial da Odometria Visual Monocular, e detalhar a implementação de um algoritmo de VO monocular. Os experimentos realizados, utilizando sequências de imagens capturadas por uma câmera rigidamente afixada no topo de uma base robótica, demonstram resultados satisfatórios. Este artigo é estruturado da seguinte maneira. A Seção 2 apresenta uma breve revisão bibliográfica, seguido de uma discussão dos elementos que compõem um algoritmo de VO. Os experimentos e resultados são apresentados na Seção 3. Finalmente, a Sessão 4 discute os resultados, trabalhos futuros e apresenta considerações sobre alguns aspectos-chave da Odometria Visual.

2 Referencial Teórico

A solução para o problema de localização com imagens tem princípio no início dos anos 80, sendo um dos primeiros trabalhos por Moravec (1980), que incidentemente também descreveu um dos primeiros detectores de cantos. A implementação utilizava uma câmera montada em um trilho, que ficava fixado no topo de um robô. O robô se movimentava e parava, fazendo a câmera deslizar, assim obtendo uma imagem em cada extremo do trilho. Portanto, este trabalho se encaixa na categoria de VO estéreo (com duas câmeras).

[1]O presente artigo foi realizado com o apoio do Edital CAPES/Pró-Alertas (88887.115590/2015-01), e em cooperação com a Hewlett Packard Brasil LTDA, com recursos provenientes da Lei de Informática (Lei nº 8.248, de 1991).

Na VO monocular, utiliza-se apenas uma câmera. Uma das mais importantes implementações desta técnica aconteceu com o trabalho de Nister et al. (2004), que foi um marco divisório nas pesquisas em VO. No artigo, foi utilizado um algoritmo RANSAC (Random Sample Consensus) modificado – com um solucionador de cinco pontos para calcular a hipótese de movimentação - para eliminação de valores extremos (*outliers*) e uma estimação de pose 3D-2D para calcular a nova pose. Mais recentemente, com a popularidade adquirida pelos métodos de SLAM e outras técnicas de navegação autônoma, surgiram trabalhos que integram essas técnicas em um sistema mais completo, como Engel et al. (2014). Neste trabalho, os autores solucionaram os problemas de estimação de postura em ambientes desconhecidos, sem a utilização de sensores possivelmente ruidosos e sem outros métodos de localização, por meio da VO monocular em um drone. Notavelmente, o sistema proposto foi disponibilizado como um pacote ROS[1], para utilização pela comunidade em outros trabalhos.

A seguir discutiremos as características gerais dos algoritmos de VO e as partes básicas de um algoritmo de VO monocular: os detectores de cantos, rastreadores de cantos e técnicas de estimação de postura.

Há diversos fatores que influenciam no resultado de uma implementação de VO, especialmente no caso monocular, como por exemplo a qualidade da imagem, referindo-se não somente à resolução mas também à maneira como a imagem é afetada por fatores externos, como mudanças bruscas de iluminamento. O grau de distorção causado pelas lentes ou por fatores construtivos da câmera também é importante, sendo que é necessário realizar a calibração da câmera para obtenção de imagens mais condizentes com o mundo real. Para a correta calibração, é necessário conhecer o modelo de projeção da câmera com que se está trabalhando. O modelo mais comum de câmera é o chamado modelo perspectivo, que assume um sistema de projeção *pinhole*, onde a imagem é formada através da intersecção dos raios de luz com o plano focal, passando através de uma abertura no centro da lente (centro de projeção). No caso deste modelo, sendo $X = [x, y, z]$ um ponto 3D no mundo real medido no sistema de coordenadas da câmera e $[u, v]$ sua projeção no plano da imagem, o mapeamento do conjunto de pontos em 3D para a imagem 2D pode ser definido como (Scaramuzza and Fraundorfer, 2011):

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = KX = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (1)$$

Na equação 1, λ é o fator de profundidade, α_u e α_v são as distâncias focais e u_0 e v_0 são as coordenadas de imagem no centro de projeção, que formam a matriz K de parâmetros intrínsecos da câmera. O objetivo da calibração é a obtenção destes parâmetros empiricamente, através de medidas conhecidas do mundo real. O método de calibração mais usual é através da análise de um plano padrão com medidas reais conhecidas (Zhang, 2000). Dois dos planos de calibração mais famosos são o tabuleiro de xadrez, onde se conhece a dimensão dos ladrilhos, e o padrão de círculos assimétricos, onde se conhece o número de círculos em duas dimensões. No caso deste projeto, foi utilizado o plano padrão xadrez para calibração, cuja matriz K é descrita na Seção 3.

Podemos enfim definir o algoritmo de VO utilizado neste artigo nos seguintes passos (Scaramuzza and Fraundorfer, 2011):

- Obter as imagens.
- Detectar pontos de interesse.
- Rastrear os pontos de interesse.
- Estimar movimentação.
- Integrar estimativa na trajetória.

2.1 Detectores de canto

Após obter as imagens corrigidas com a calibração, deve-se realizar a detecção de pontos de interesse (também conhecidos como *feature points*). Através dos pontos é possível estimar o movimento, e a qualidade e quantidade destes pontos irão determinar a precisão do algoritmo. Os cantos normalmente são escolhidos como feature points, pois é mais fácil de se detectar seus correspondentes em uma segunda imagem, e porque são mais salientes que outros tipos de pontos de interesse (por exemplo, bordas). A seguir serão discutidos os detectores de Harris e Shi-Tomasi utilizados neste trabalho.

Os detectores de canto clássicos (por exemplo, Moravec) consideram uma janela local em uma imagem e determinam as variações de intensidade que resultam da movimentação desta janela em várias direções (Harris and Stephens, 1988), para qual há três resultados possíveis:

1. Se a janela local não encontra mudanças significativas em intensidade, todas as movimentações resultam em aproximadamente a mesma variação.
2. Se a janela encontra uma borda, uma movimentação ao longo da borda resultará em uma pequena variação. Uma movimentação na direção perpendicular à borda resultará em uma grande variação.

[1] `http://wiki.ros.org/tum_ardrone`

3. Se a janela encontrar um canto ou ponto isolado, todas as movimentações resultarão em grandes variações. O canto pode então ser detectado encontrando-se o ponto em que a mudança mínima produzida por qualquer movimentação é grande.

É possível encontrar os cantos em uma imagem com a relação matemática:

$$P_{u,v} = \sum_{i,j} w_{i,j}(I_{u+i,v+j} - I_{i,j})^2 \qquad (2)$$

onde $I \in \mathbb{R}^{n,n}$ é uma matriz de números reais em que cada posição *(i,j)* descreve a intensidade do respectivo pixel, $w_{i,j}$ é a função-janela e $P_{u,v}$ é a variação produzida por uma movimentação da janela na direção *(u,v)* (Harris and Stephens, 1988).

O detector de Harris é um incremento ao detector de Moravec, onde são resolvidos três problemas. Primeiramente, o detector de Moravec gera outliers, devido à sua função-janela binária. Para solucionar este problema, Harris utiliza uma função-janela gaussiana, que dá um aspecto circular à janela local. Em segundo lugar, o detector de Moravec é anisotrópico - ou seja, suas propriedades são dependentes da direção de deslocamento da janela. Harris corrige este problema com uma expansão analítica no entorno da origem da janela. Finalmente, o último problema é que há uma resposta muito imediata a bordas, já que apenas o valor mínimo de $P_{u,v}$ é considerado em uma janela local. Portanto, utiliza-se a variação de $P_{u,v}$ com a direção da movimentação que, considerando uma movimentação pequena em *x,y*, pode ser escrita como:

$$P_{x,y} = (x,y)M(x,y)^T \qquad (3)$$

Onde a matriz M contém os produtos vetoriais das variações com a função-janela. Se considerarmos os autovalores de M como λ_1 e λ_2, podemos também definir:

$$S_{har} = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 \qquad (4)$$

Onde k define o limiar para que uma região seja considerada "plana" (sem cantos ou bordas) (Harris and Stephens, 1988). A Equação 4 define o "escore", que relaciona a presença de cantos e bordas à regiões-janela de uma imagem. Assim, para os valores de S_{har}:

- $S_{har} = 0$: Região plana.
- $S_{har} < 0$: Região de borda.
- $S_{har} > 0$: Região de canto.

O detector de Shi-Tomasi é um incremento ao detector de Harris. A variável S_{har} vista na Equação 4 é dependente da determinante e do traço da matriz M. No detector de Shi-Tomasi, toma-se como escore apenas o valor mínimo entre os autovalores de M (Shi and Tomasi, 1993):

$$S_T = min(\lambda_1, \lambda_2) \qquad (5)$$

Assim, se S_T está acima de um valor limite, aquela é uma região de canto. É possível visualizar isto na Figura 1, onde o ponto (λ_1, λ_2) é uma região de canto se ambos os autovalores estiverem acima do valor limite.

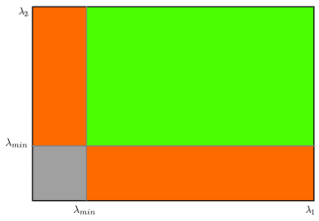


Figura 1: Espaço de autovalores do detector de Shi-Tomasi

2.2 Rastreamento de Features

Após realizada a etapa de detecção, é necessário realizar o cálculo do fluxo ótico dos pontos de interesse - isto é, rastreá-los em uma sequência de imagens. Neste trabalho foi utilizado o rastreador de Kanade-Lucas-Tomasi (rastreador KLT) com representação piramidal (Bouguet, 2000).

O problema do cálculo do fluxo ótico consiste em realizar uma análise no entorno da localização conhecida de um ponto de interesse, procurando por este mesmo ponto em uma imagem seguinte. Mais detalhadamente, considera-se um ponto $u = (u_x, u_y)$ em uma primeira imagem. O objetivo do rastreamento é encontrar o mesmo ponto em uma segunda imagem, e podemos descrever o vetor $d = (d_x, d_y)$ como sendo o fluxo ótico daquele ponto entre as duas imagens. Porém, graças ao *Aperture Problem* (McDermott et al., 2001), é necessário realizar uma análise da vizinhança do ponto u na segunda imagem. Define-se para tanto w_x e w_y como inteiros, onde o fluxo d será definido como o vetor que minimiza a função residual ϵ:

$$\epsilon(d_x, d_y) = \sum_{x=u_x-w_x}^{u_x+w_x} \sum_{y=u_y-w_y}^{u_y+w_y} (I_{(x,y)} - J_{(x+d_x,y+d_y)})^2 \qquad (6)$$

Seguindo esta definição, a função de similaridade é medida em uma vizinhança de tamanho $(2w_x + 1) \times (2w_y + 1)$. Esta é conhecida também como janela de integração.

Dois fatores chave que determinam a performance de um rastreador são a precisão e a robustez. A precisão será dependente do tamanho da janela de integração escolhida, que deve intuitivamente ser pequena para não descartar os

detalhes nas imagens. A robustez depende da sensibilidade do rastreador quanto a mudanças em luminosidade, velocidade de movimentação ou tamanho de imagem, por exemplo. Intuitivamente, é preferível utilizar uma janela de integração grande para computar movimentações grandes, o que vai de contrapartida ao requisito de precisão (Bouguet, 2000). Há então uma relação de custo-benefício na escolha da janela de integração. O rastreador KLT modificado proposto por Bouguet (2000) utiliza uma representação piramidal para solucionar este problema. Define-se a representação piramidal de uma imagem genérica I de tamanho $n_x \times n_y$, onde $I^0 = I$, que é a imagem original na maior resolução. Define-se então I^1 a partir de I^0, I^2 a partir de I^1 e assim por diante, até I^L a partir de $I^{L\text{-}1}$. As imagens em níveis crescentes são construídas com resoluções decrescentes.

A questão da quantidade de níveis que uma representação deve ter é definida empiricamente. Normalmente não vale a pena ir além do quarto nível por causa das dimensões reduzidas da imagem.

Enfim, retornando ao objetivo inicial do rastreador, podemos derivar o vetor d. Sendo $q^L = (q_x^L, q_y^L)$ as coordenadas do ponto q nas imagens piramidais I^L, os vetores de q^L são definidos como:

$$q^L = \frac{q}{2^L} \tag{7}$$

A operação na equação 7 é aplicada para ambas as coordenadas de q independentemente. O algoritmo de rastreamento piramidal então é (Bouguet, 2000):

1. Computar o fluxo ótico $v = q + d$ no nível L mais alto.
2. Propagar o resultado da operação anterior para o nível L - 1 na forma de uma estimativa inicial para o deslocamento no nível L - 1.
3. Refinar o fluxo ótico no nível L - 1, que é propagado para o nível L - 2.
4. Repetir os itens 2 e 3 até chegar ao nível L = 0 (a imagem original).

Vale notar que, dado um tempo de movimentação, os pontos de interesse rastreados não estarão mais no campo de visão da câmera, sendo necessário definir novos pontos de interesse a ser rastreados.

2.3 Estimação de Movimentação

A quarta etapa do algoritmo de VO é a estimação de movimentação, utilizando os pontos de interesse obtidos através da detecção e rastreamento dos cantos. Ela consiste na definição da rotação e translação do sistema de coordenadas da câmera na imagem I_k referente ao sistema de coordenadas na imagem anterior, sendo assim efetivamente a movimentação da câmera entre duas imagens. Esta movimentação pode ser obtida através da restrição epipolar. Para entender esta restrição, considere um ponto X em três dimensões no mundo real. Captura-se uma imagem em uma postura C_{k-1} que contém X, com seu ponto correspondente na imagem definido como $\widetilde{p}$. A câmera movimenta-se e captura outra imagem em uma postura C_k que contém X, desta vez com o ponto correspondente como $\widetilde{p}'$. Assim, $\widetilde{p}$ e $\widetilde{p}'$ são dois pontos com coordenadas de imagem distintas que correspondem ao mesmo ponto X do mundo real. A restrição epipolar descreve a diferença entre $\widetilde{p}$ e $\widetilde{p}'$ através da matriz essencial E_{mat}, que contém a translação $\hat{t}$ e rotação R entre os dois pontos. A restrição é definida como $\widetilde{p}E\widetilde{p}' = 0$.

A matriz E_{mat} pode ser encontrada com no mínimo 5 correspondências (considerando 6 graus de liberdade). O algoritmo de RANSAC de Nister et al. (2004) tornou-se padrão para a obtenção da matriz essencial com correlação 2D-2D. A partir dessa matriz, é possível extrair as matrizes de rotação e translação através de uma decomposição por valor singular (do inglês, SVD). As matrizes de rotação e translação são definidas como (Scaramuzza and Fraundorfer, 2011):

$$R = U(\pm W^T)V^T \tag{8}$$

$$\hat{t} = U(\pm W)SU^T \tag{9}$$

Sendo que W^T e $\hat{t}$ são, respectivamente:

$$W^T = \begin{bmatrix} 0 & \pm 1 & 0 \\ \mp 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{10}$$

$$\hat{t} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \tag{11}$$

Notavelmente, há quatro possíveis combinações de soluções para as matrizes R e $\hat{t}$. Para selecionar a combinação adequada, deve-se realizar a triangulação de um ponto arbitrário, medindo sua rotação e translação reais para então comparar e identificar a combinação certa empiricamente. A matriz R e o vetor t são então inseridos na forma da transformação rígida:

$$T_{k,k-1} = \begin{bmatrix} R_{k,k-1} & t_{k,k-1}^T \\ 0 & 1 \end{bmatrix} \tag{12}$$

Esta transformação é então armazenada em C_k, que é vetor de posições da câmera nos momentos k. Este vetor é o objetivo final da odometria visual, contendo a movimentação completa da câmera no ambiente.

3 Resultados

Para realização dos experimentos e obtenção dos resultados, foram implementados: um algoritmo de captura de sequencias de imagens através de uma câmera USB; um algoritmo de odometria visual com detectores de cantos (Harris e Shi-Tomasi), o rastreador KLT e o método de estimação 2D-2D.

Para as tarefas de processamento de imagem, foram utilizadas as bibliotecas OpenCV na linguagem C++. Os experimentos foram realizados com sequências de imagens obtidas a 30 quadros-por-segundo em um ambiente interno, com a base robótica TurtleBot 2 e a câmera PlayStation Eye afixada ao topo da base. Percorreu-se um retângulo de 16 por 30 metros manualmente, com o robô em modo de teleoperação. As imagens deste trajeto foram capturadas e o algoritmo de VO foi utilizado posteriormente (*offline*). O algoritmo foi testado em um computador com sistema operacional Ubuntu 12.04 e processador Intel i7-3632QM, com 8 núcleos de 2,2 GHz. A calibração da câmera foi realizada utilizando uma função padrão do OpenCV juntamente com o padrão xadrez, resultando na matriz K:

$$K_{xadrez} = \begin{bmatrix} 541,218 & 0 & 319,5 \\ 0 & 541,218 & 239,5 \\ 0 & 0 & 1 \end{bmatrix} \quad (13)$$

Com os detectores de Harris e Shi-Tomasi, obteve-se quantidades distintas de pontos de interesse, sendo que o detector de Shi-Tomasi resultou na maior quantidade. Em contrapartida, o detector de Harris resultou em um tempo computacional total menor. Os tempos computacionais são mostrados na Tabela 1, referentes à execução completa do programa principal que inclui as partes de detecção, rastreamento e estimação, sem nenhum tipo de paralelismo em sua implementação. A Tabela também indica a quantidade média de cantos detectados por cada detector.

Tabela 1: Tabela comparativa de tempos computacionais para os dois detectores

	Tipo de detector	
	Harris	**Shi-Tomasi**
Tempo médio de processamento do programa principal (T_{prog})	0,029 s	0,046 s
Tempo médio de processamento de um par de imagens (T_{par})	$2,85 \times 10^{-5}$ s	$4,42 \times 10^{-5}$ s
Número médio de cantos	124	688

Os desvios-padrão entre as medidas de tempo são, respectivamente, $\sigma_{prog} = 0,012$ e $\sigma_{par} = 1,11 \times 10^{-5}$ segundos, o que indica pequena variabilidade entre os dois detectores em questão de custo computacional.

O que mais influenciou no sucesso do algoritmo foi a qualidade dos pontos de interesse, que varia entre os detectores. O detector de Harris acaba identificando aglomerações de pontos, enquanto que pontos mais espalhados no ambiente são mais adequados a esta aplicação. É possível ainda melhorar a performance através de modificações nos parâmetros dos detectores (por exemplo, escolher uma função-janela diferente). A Figura 2 mostra a estimação de uma trajetória em circuito utilizando o detector de Harris, enquanto que a Figura 3 mostra a mesma trajetória com o detector de Shi-Tomasi.

O melhor resultado obtido foi o da Figura 3, com o detector de Shi-Tomasi e plano de calibração xadrez, pois este se assemelha mais com o trajeto real em formato retangular. Nota-se que a postura final não é perfeitamente condizente com a postura inicial, graças ao efeito de *drift*, que é o acumulo de pequenos erros de medição. Outra possível causa de falha é a dificuldade de garantir a rigidez da câmera no corpo da base robótica. A escolha da frequência de quadros (FPS) também é importante na redução do *drift*, pois menos quadros na sequência significa menor erro. A redução do FPS também influencia na velocidade estimada do robô, pelo aumento do fluxo óptico. Uma boa taxa de FPS para esta implementação é de 15 FPS, descoberta empiricamente considerando a velocidade do robô de 0,34 m/s.

Um exemplo dos pontos detectados com o detector de Shi-Tomasi pode ser visto na Figura 4a, enquanto que a Figura 4b mostra os pontos detectados com o detector de Harris. Comparando as imagens, é possível ver a diferença entre os detectores quanto à quantidade dos pontos detectados. Quanto mais pontos de interesse detectados, maior é a precisão do algoritmo aqui implementado.

Finalmente, um vídeo do algoritmo em ação encontra-se na Seção de projetos do website do Grupo de Automação e Controle de Sistemas (GACS)[2] da Pontifícia Universidade Católica do Rio Grande do Sul.

4 Conclusões

O método de VO mostra-se robusto para implementações em ambientes internos. Dentre as vantagens do método, pode-se citar que ele não sofre com instabilidade de terreno, como acontece com a odometria por rodas. Porém, a eficácia da VO é altamente dependente de muitos fatores como mudanças de iluminação, quantidade de pontos de interesse, mudanças repentinas nas imagens (por exemplo, pessoas se movimentando), entre outros. Também é de vital importância selecionar o detector certo para cada implementação, assim como a calibração correta para determinada câmera, além da frequência de captura.

[2] http://www.feng.pucrs.br/~gacs/index.php?_lab=&_lang=br&_pg=projetos

Figura 2: Resultado da Odometria Visual utilizando o detector de Harris a 15 FPS e calibração com plano xadrez

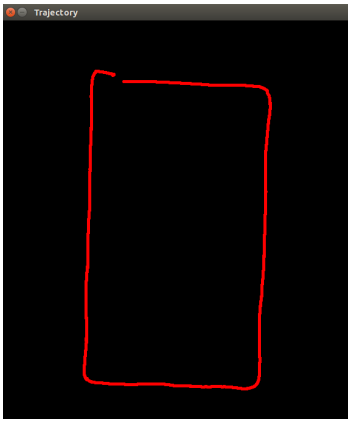
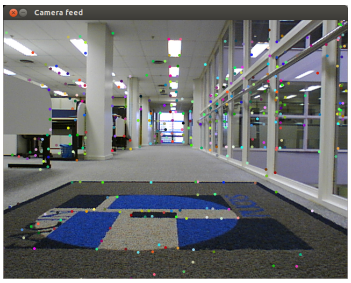
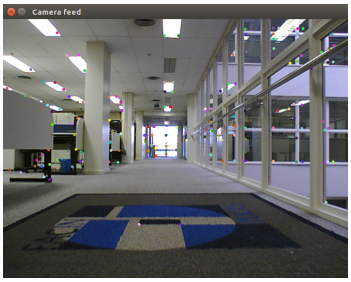


Figura 3: Resultado da Odometria Visual utilizando o detector de Shi-Tomasi a 15 FPS e calibração com plano xadrez



(a)



(b)

Figura 4: Exemplo de pontos de interesse detectados com os detectores de Shi-Tomasi e Harris respectivamente

Como trabalhos futuros, é possível: implementar o algoritmo utilizando detectores de região e descritores, para análise de desempenho em ambientes externos sem muitos cantos; disponibilizar a implementação para dispositivos móveis, como celulares ou sistemas embarcados; comparar esta implementação com outros métodos de odometria, para investigar as variáveis que afetam cada técnica e as diferenças entre elas. Existem ainda maneiras de realizar otimizações das poses obtidas, para minimizar a propagação do erro de estimação. Pode-se utilizar, por exemplo, uma técnica chamada *Windowed Bundle Adjustment*, que consiste em realizar uma otimização local em uma janela de m posições.

Referências

Bouguet, J. (2000). Pyramidal implementation of the lucas kanade feature tracker, *Intel Corporation, Microprocessor Research Labs* .

Engel, J., Sturm, J. and Cremers, D. (2014). Scale-aware navigation of a low-cost quadrocopter with a monocular camera, *Robotics and Autonomous Systems (RAS)* **62**.

Harris, C. and Stephens, M. (1988). A combined corner and edge detector, *In Proc. of Fourth Alvey Vision Conference*, pp. 147–151.

McDermott, J., Weiss, Y. and Adelson, E. H. (2001). Beyond junctions: Nonlocal form constraints on motion interpretation, *Perception* **30**(8): 905–923.

Moravec, H. (1980). Obstacle avoidance and navigation in the real world by a seeing robot rover, *Technical Report CMU-RI-TR-80-03*.

Nister, D., Naroditsky, O., Bergen, J. and Sarnoff Corporation (2004). Visual odometry, *Proc. Int. Conf. Computer Vision and Pattern Recognition*, pp. 652–659.

Scaramuzza, D. and Fraundorfer, F. (2011). Visual odometry: Part 1 - the first 30 years and fundamentals, *IEEE Robotics and Automation Magazine* **18**(4).

Shi, J. and Tomasi, C. (1993). Good features to track, *Technical report*, Ithaca, NY, USA.

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