

Projeto Demonstrativo 03 - Múltiplas Vistas

Thúlio Noslen Silva Santos (14/0164090)
Departamento de Ciéncia da Computaçao
Universidade de Brasília
Brasília, Brasil
thulionoslen_@hotmail.com

Resumo—Um problema biologicamente inspirado na área de visão computacional é o de extração de informações em 3D sobre o mundo a partir de múltiplas vidas. Neste trabalho, será tratado o problema de visão estéreo, e métodos de calibração e geometria epipolar serão utilizados a fim de criar mapas de disparidade e profundidade e conseguir medir o tamanho de objetos a partir das vistas de duas câmeras.

Index Terms—visão computacional, OpenCV, múltiplas vistas, visão estéreo

I. INTRODUÇÃO

A partir de somente uma imagem, é possível extrair informações sobre o mundo se os parâmetros intrínsecos e extrínsecos da câmera forem conhecidos. Para sistemas de visão estéreo, informações sobre o mundo podem ser extraídas sem que a calibração extrínseca seja realizada. Para isto, este tipo de sistema usa relações baseadas em geometria epipolar.

A. Geometria epipolar

A geometria epipolar descreve as relações entre o ponto no mundo e a projeção deste ponto nos planos de imagem das duas câmeras num sistema de visão estéreo. Um exemplo deste tipo de geometria pode ser conferido na figura 1, retirada de [3].

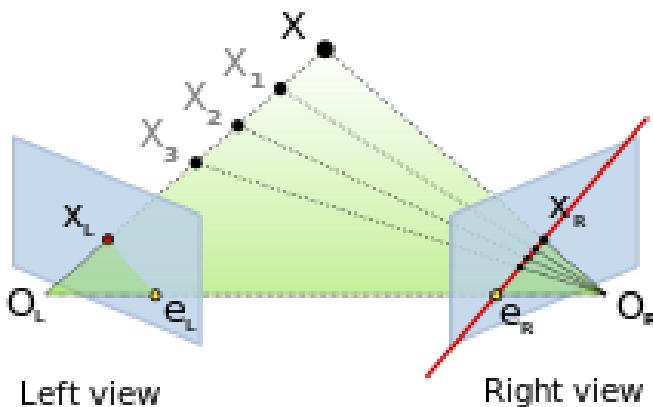


Figura 1. Exemplo de geometria epipolar.

Um dos principais resultados obtidos a partir desta geometria é a relação triangular que existe entre a distância entre as

câmeras b , a distância focal f , a disparidade, calculada como $d = x_l - x_r$, e a profundidade Z :

$$Z = \frac{bf}{x_l - x_r} \quad (1)$$

Para X e Y valem as seguintes relações:

$$X = \frac{b(x_l + x_r)}{2(x_l - x_r)}, Y = \frac{b(y_l + y_r)}{2(x_l - x_r)} \quad (2)$$

Com isto, sabendo-se f e b , pode-se montar um mapa de disparidade a partir de uma imagem de cada câmera do sistema estéreo e a partir deste mapa de disparidade montar um mapa de profundidade. Como geralmente os mapas almejados são densos, podem ser usados métodos de template matching para determinar as correspondências entre todos, ou quase todos, os pixels das imagens.

B. Retificação

Para que os resultados acima sejam válidos, é necessário que os planos de imagem das duas câmeras sejam paralelos. Se este não for o caso, é necessário que as imagens sejam retificadas, ou seja, que sejam mapeadas para um mesmo plano paralelo. Para isto, é necessário conhecer a relação projetiva entre os dois planos. Este conhecimento pode ser adquirido a partir das matrizes de projeção das duas câmeras, ou, assim como anteriormente, pode ser derivado a partir das próprias imagens as câmeras.

Ao invés de utilizar as matrizes de projeção, pode-se derivar a relação entre os dois planos de imagem das câmeras diretamente a partir de princípios de geometria epipolar. A relação projetiva entre os dois planos pode ser expressa com uma matriz F , chamada de matriz fundamental, que é tal que:

$$x'^T F x = 0 \quad (3)$$

Em que x é o ponto no mundo X projetado na primeira câmera e x' é a projeção na segunda câmera. Observe que se a correspondência de pontos de uma imagem para outra for conhecida para um número suficiente de pontos, a equação acima pode ser resolvida para F usando métodos de otimização linear.

C. Reconhecimento de features

Para determinar a matriz F , é necessário um número mínimo de correspondências entre pontos das duas imagens. Os mesmos métodos de template matching usados para calcular

o mapa de disparidade não são boas escolhas aqui, porque não são necessários todos os pontos e os métodos de template matching não são muito precisos. Para determinar quais são estes pontos, pode-se usar técnicas como o de reconhecimento de features, usando descritores como ORB e SIFT.

Uma vez que estes pontos sejam determinados, a matriz F pode ser determinada, e com estas duas informações em mãos, o mapeamento de retificação pode ser calculado, mesmo que os parâmetros de calibração não sejam conhecidos. O método, em linhas gerais, se baseia em alinhar os features encontrados de forma que as linhas epipolares nas duas imagens sejam horizontais.

II. MATERIAIS E METODOLOGIA

A. Materiais

O programa desenvolvido deve atender aos seguintes requisitos:

- Gerar um mapa de profundidade para imagens já retificadas e alinhadas;
- Retificar imagens que foram capturadas por duas câmeras que não necessárias estão alinhadas;
- Permitir que sejam feitas medidas de objetos na imagem e calcular a medida no mundo real.

Foi utilizado um computador com as seguintes configurações:

- Ubuntu 16.04.03 LTS 64-bits;
- OpenCV 3.4.0;
- numpy 1.14.2;
- Python 3.5

B. Metodologia

Foram utilizados tutoriais de Python, numpy, OpenCV, template matching, feature matching e geometria epipolar na elaboração do projeto. A câmera utilizada foi a câmera frontal de um Moto G4 Plus. Mais informações sobre a implementação do trabalho pode ser conferida no código fonte e no arquivo README.md.

Observe que, no caso deste projeto, não foram utilizadas duas câmeras, e sim uma câmera que se movimentava entre dois pontos. Em teoria, a resolução do problema se dá da mesma forma.

C. Requisito 1

Para a computação de um mapa de profundidade para o primeiro conjunto de imagens, que já está retificada e já se conhecem a distância focal e a distância entre as câmeras, foram implementados dois métodos: template matching para encontrar pontos de correspondência e uma implementação pronta de block matching no OpenCV.

Para o método "manual" que utiliza template matching, temos o seguinte. Primeiro, determinam-se parâmetros como tamanho da janela e a fração da largura da imagem em que o ponto será procurado. Em seguida, para cada centro de janela possível na imagem, um template é formado com a janela e é deslizado horizontalmente na imagem para determinar o melhor encaixe. Com isto, a melhor correspondência é gravada

e a disparidade vai sendo calculada. Ao final, quando o mapa de disparidade estiver pronto, um filtro é utilizado para ignorar os outliers dentro do mapa. Com isto, elimina-se alguns "falsos positivos", ou pontos com disparidade muito altas que foram mapeados no lugar errado. Em seguida, o algoritmo aplica a equação 1 para determinar o mapa de profundidade. Para armazenamento, a disparidade é normalizada para a faixa 0-255, em que NaN indica que não foi possível determinar uma correspondência, e o logaritmo do mapa de profundidade é normalizado para a faixa 0-255, em que 0 indica que não foi possível determinar uma correspondência.

O método "automático", que utiliza um método de block matching pronto do OpenCV que já calcula a disparidade, consiste em determinar a disparidade com o algoritmo do OpenCV e determinar o mapa de profundidade conforme acima.

1) *Requisito 2*: Para as imagens "imperfeitas", em que os planos não estão alinhados, primeiro o programa precisa retificar as duas imagens e aplicar os mesmos passos que foram aplicados ao conjunto de imagens "perfeitas". Para retificar as imagens, é necessário determinar um conjunto de correspondências preciso entre as duas imagens. Primeiro, o programa instancia um descritor de features. Os detectores ORB e SIFT foram implementados, mas somente um pode ser escolhido durante a execução. Por padrão, somente o descrito SIFT foi utilizado, por razões que serão detalhadas mais à frente. Depois que um dos dois gera um conjunto de pontos adequado, a matriz F é calculada com uma função específica para isso do OpenCV. Em seguida, mais funções prontas do OpenCV são usadas para determinar a homografia de retificação das imagens e para aplicar esta homografia.

Assim, o último passo é determinar a distância focal e a distância entre as câmeras antes de computar o mapa de profundidade. Para determinar a distância focal, o programa calcula a média entre f_x e f_y , que são parâmetros da matriz de calibração K . A matriz K foi reaproveitada de projetos anteriores, visto que a câmera é a mesma. Para determinar a distância entre as câmeras, foi realizada uma calibração extrínseca com o tabuleiro sendo capturado pelas duas câmeras. A matriz de parâmetros extrínsecos foi determinada para cada câmera, e, com isso, foi estimada a distância entre os centros das câmeras.

D. Requisito 3

Para o último requisito, para medir a distância no mundo com distâncias em pixels, foram aplicadas as equações 1 e 2 depois de aplicar às devidas homografias ao pixels escolhidos. Assume-se que o primeiro clique em cada imagem representa pixels correspondentes, assim como o segundo clique. Para ambos os cliques, o vetor $[XYZ]^T$ é calculado, e a norma da diferença entre os vetores dá o tamanho do objeto.

III. RESULTADOS

Todos os comandos usados para obter os resultados podem ser conferidos no arquivo README.md.

A. Requisito 1

Nas figuras 2, 3, 4 e 5 podem ser conferidos os mapas de profundidades gerados pelo programa para os dois pares de imagens.



Figura 2. Primeira imagem.

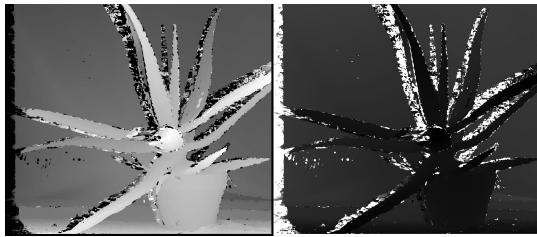


Figura 3. Mapas de disparidade e profundidade da primeira imagem, na esquerda e direita, respectivamente.



Figura 4. Segunda imagem.

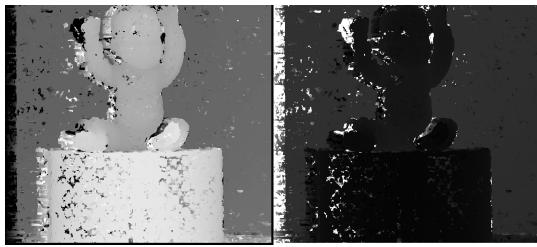


Figura 5. Mapas de disparidade e profundidade da segunda imagem, na esquerda e direita, respectivamente.

B. Requisito 2

Nas figuras 6, 7 e 8 pode ser conferido o resultado dos mapas para as imagens capturadas pela câmera do celular. No terminal, observou-se $Z_{min} = 641.64mm$ e $Z_{max} : 2673.50mm$.



Figura 6. Imagem capturada pelo celular.

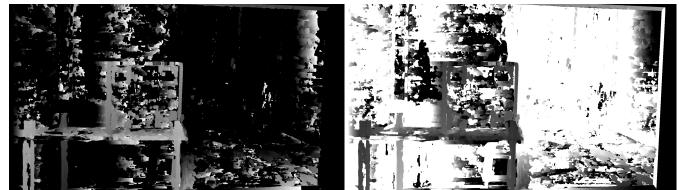


Figura 7. Mapas de disparidade e profundidade da imagem capturada pelo celular, na esquerda e direita, respectivamente.

C. Requisito 3

O resultado de uma medição para uma imagem capturada no celular pode ser vista na figura 9.

Foram obtidos os seguintes valores: $f = 655.60pixels$, $b = 122.34mm$, $dist = 891.76mm$.

IV. DISCUSSÃO E CONCLUSÕES

Alguns dos testes não couberam no relatório, então o programa foi projetado de forma que todos as observações abaixo podem ser replicadas usando alguns comandos específicos encontrados no arquivo README. Também segue no README um link para um repositório no GitHub onde se encontram algumas imagens que ajudaram a realizar o fine-tuning experimental dos parâmetros.

A. Requisito 1

Observe que as imagens ficaram bastante próximas do esperado, tanto de disparidade quanto de profundidade, apesar de a profundidade estar mais difícil de ser observar, por estar bastante escura. Foi utilizada uma "heurística" para ajudar no matching do algoritmo: o espaço de busca horizontal é de 25% em torno do centro do template atual. Este foi um valor determinado experimentalmente e foi escolhido de forma que não causasse nenhum objeto próximo (i.e. com alta disparidade) sem ter seu match encontrado.



Figura 8. Mapa de profundidade da imagem capturada pelo celular, gerado usando SGBM do OpenCV.

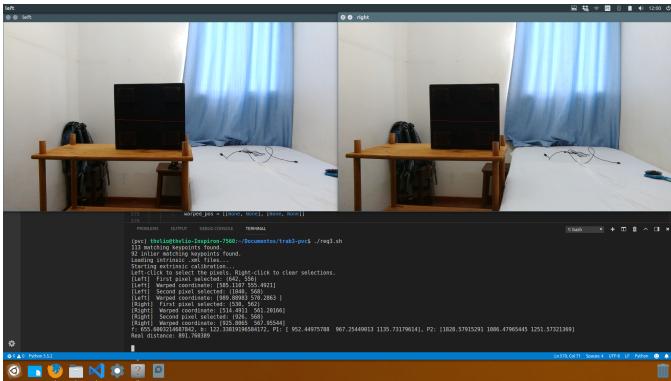


Figura 9. Resultado de medida de largura.

B. Requisito 2

A primeira coisa a se notar é que existe muito ruído nas imagens. Algumas silhuetas são visíveis, sendo que somente a cortina está razoavelmente mapeada. Isto possivelmente acontece porque a textura da cortina não é tão lisa quanto a da parede branca ao fundo, e não é composta por padrões que se repetem.

Nas imagens capturadas pelo celular, a roupa de cama foi trocada e os travesseiros retirados da imagem, pois ambos eram tecidos com padrões que se repetiam, e isto fazia com que os descritores de features se aglomerassesem nestas regiões. Este problema pode ser observado se o descritor ORB for utilizado ao invés do SIFT. Com o ORB, este problema é muito mais acentuado. Como todos os pontos se aglomeravam em uma ou duas regiões, a retificação se tornava muito distorcida. Isto pode ser observado se o código for executado com o descritor ORB e com a opção "show-matches" ativada. Nas imagens nas pastas `discard1` e `discard2`, os descritores tem muita dificuldade de gerar pontos adequados para a retificação.

Veja que na figura 8, em que foi usado o SGBM do OpenCV, a profundidade está menos normalizada, permitindo ver melhor a silhueta dos objetos, enquanto que a profundidade na figura 7 atinge o máximo na parede, fazendo com que seja difícil discernir entre a cortina, a mochila e a parede.

C. Requisito 3

Veja que f concorda com os valores do projeto anterior, e b é próximo do valor correto, de 120 mm. Todas as imagens foram capturadas com este baseline. Já para a distância, temos um valor que está longe do esperado, que era de 350 mm. Isto pode estar acontecendo por uma falha na implementação, uma vez que as imagens estão sendo reduzidas pela metade para que caibam na tela.

D. Requisito 4

Existem várias sugestões de melhoria. Uma delas é na filtragem dos outliers realizada logo após a computação do mapa de disparidade. Ela trabalha com uma média e desvio padrão geral da imagem, mas talvez fosse mais interessante que este filtro trabalhasse com regiões na imagem de cada vez.

Outra melhoria poderia ser o uso mais sério do SGBM do OpenCV. Ele foi usado somente para comparar o algoritmo criado com um algoritmo já estabelecido, mas ele poderia também ter sido usado para gerar o mapa de profundidade.

O tamanho da janela define o quanto de ruído e quanta precisão o mapa terá. Quanto menor a janela, melhores são os contornos dos objetos, mas pode-se observar muito ruído. Nas pastas `test1` e `test6`, o subscrito '`_w`' no nome do arquivo indica o tamanho de janela escolhido. Pode-se notar, principalmente nas imagens em `test1`, que quanto mais se aumenta a janela, mais o ruído some, mas os contornos começam a perder a forma.

Para testes feitos com "heurística" de busca horizontal, temos as pastas `test1`, `test5` e `test7`, onde o subscrito '`_n`' indica que a busca foi reduzida para uma janela de $width/n$ em torno do centro do template.

Conforme já mencionado, as paredes lisas, ou com textura muito uniforme, acabam gerando problemas no template matching, uma vez que a disparidade observada será quase aleatória. Observe que existe uma espécie de ruído onde a textura é uniforme na figura 7: na tábua preta, no chão de madeira e na parede branca.

Por fim, o algoritmo de template matching do OpenCV é bastante lento. Isto acontece porque não há nenhuma otimização específica que leva em conta que a maior parte dos cálculos realizados na janela anterior podem ser reutilizados. Este atraso já não acontece com o SGBM, que já é preparado para este tipo de cálculo.

REFERÊNCIAS

- [1] David A. Forsyth, Jean Ponce, "Computer Vision: A Modern Approach," Second Edition, Prentice Hall, 2011.
- [2] Richard Hartley, Andrew Zisserman, "Multiple View Geometry in Computer Vision," Second Edition, Cambridge University Press, 2004.
- [3] Wikipedia, "Epipolar geometry - Wikipedia," https://en.wikipedia.org/wiki/Epipolar_geometry. Acessado em 03/05/2018.
- [4] OpenCV, "Camera Calibration and 3D Reconstruction — OpenCV 3.0.0-dev documentation," https://docs.opencv.org/3.0-beta/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html. Acessado em 03/05/2018.
- [5] OpenCV, "Epipolar Geometry — OpenCV 3.0.0-dev documentation," https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_calib3d/py_epipolar_geometry/py_epipolar_geometry.html. Acessado em 03/05/2018.
- [6] OpenCV, "Depth Map from Stereo Images — OpenCV 3.0.0-dev documentation," https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_calib3d/py_depthmap/py_depthmap.html. Acessado em 03/05/2018.
- [7] OpenCV, "Template Matching — OpenCV 3.0.0-dev documentation," https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_template_matching/py_template_matching.html. Acessado em 03/05/2018.
- [8] OpenCV, "Feature Matching — OpenCV 3.0.0-dev documentation," https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_matcher/py_matcher.html. Acessado em 03/05/2018.