

000

001 Projeto Demonstrativo 3 - Câmeras Estéreo

002

003 Natalia Oliveira Borges
004 natioliveira97@hotmail.com

005 Departamento de Ciência da
006 Computação
007 Universidade de Brasília
008 Campus Darcy Ribeiro, Asa Norte
009 Brasília-DF, CEP 70910-900, Brazil,

010

Matrícula:
16/0015863

011

Lívia Fonseca
liviagcf@gmail.com

012

Matrícula:
16/0034078

013

0

014

015 Abstract

016

017 Câmeras estereoscópicas são câmeras que possuem mais de uma lente. Assim cap-
018 turam a mesma cena em duas imagens, uma um pouco deslocada da outra, permitindo a
019 reconstrução tridimensional da cena. Esse mesmo efeito pode ser obtido usando duas ou
mais câmeras que capturam a mesma cena.

020

021 Esse projeto consiste na construção dos mapas de disparidade e de profundidade
022 de três cenas baseada em duas fotos em estéreo. E, usando esses mapas, determinar o
volume de um objeto em uma das cenas.

023

024 1 Introdução

025

026 Câmeras fotográficas são uma das muitas formas de se obter informações do ambiente que
027 nos cerca. Contudo, essa informação é limitada, pois câmeras convertem imagens do mundo
028 3D para o 2D. Uma forma de recuperar as informações relativas a profundidade é usando
029 imagens de câmeras em estéreo.

030 Para conseguir reconstruir a imagem em 3D, são necessários dois mapas, um de dispari-
031 dade e, a partir dele, um de profundidade. O mapa de disparidade é uma matriz que armazena
032 a distância entre os pixels das imagens que correspondem ao mesmo ponto no mundo real. Já
033 o mapa de profundidade é definido usando os valores de disparidade, a *baseline* e a distância
034 focal e armazena os valores de profundidade de uma cena em 3D [1].

035 Quando temos imagens retificadas, ou seja, a posição y dos pixels na imagem não muda
036 de uma imagem para a outra, o trabalho de encontrar a correspondência de pixels entre uma
037 imagem e a outra é reduzido, já que só temos que procurar o pixel na mesma linha. Para isso,
038 usamos um algoritmo baseado na soma de diferenças absolutas, que será explicado em 2.1.

039 Quando as imagens não estão retificadas existem algumas abordagens possíveis. Pode-se
040 retificar a imagem conhecendo seu parâmetros intrínsecos e extrínsecos e assim calcular a
041 homografia de uma para a outra [2]. Ou pode-se encontrar a matriz fundamental e a partir
042 dela as linhas epipolares.

043

© 2018. The copyright of this document resides with its authors.

044

It may be distributed unchanged freely in print or electronic forms.

045

045 ºO trabalho foi feito em conjunto, tendo ambas as partes contribuído para o desenvolvimento de cada requisito
e para a elaboração do relatório

A matriz fundamental de uma imagem é calculada a partir de pontos correspondentes de duas imagens. Obtendo a matriz fundamental, para qualquer ponto em uma imagem é possível calcular uma linha na outra imagem em que esse ponto estará localizado, essas linhas são chamadas linhas epipolares [4].

O objetivo final desse projeto é usar informações obtidas por duas câmeras para estimar a posição 3D de uma cena. E, usando o mapeamento em 3D, estimar o volume de um objeto.

2 Metodologia

Esse trabalho foi feito em Linux Xubuntu versão 18.04 e em Linux Ubuntu versão 16.04, ambos usando Opencv versão 3.2.0. O código foi feito em C++.

2.1 Requisito 1

2.1.1 Encontrando o mapa de disparidade

Nesse requisito, devemos encontrar o mapa de disparidade entre duas imagens estéreo, previamente retificadas. Para isso devemos encontrar correspondências entre uma imagem e a outra. Uma forma de verificar correspondência pixel a pixel é comparar uma pequena janela da imagem da esquerda com a imagem da direita e encontrar, nessa imagem, a janela mais parecida. Como a imagem já está retificada, devemos procurar o pixel na mesma linha em que ele se encontra, poupando esforço computacional.

Usando a soma de diferenças absolutas , podemos encontrar qual pixel na imagem da direita corresponde ao pixel da esquerda, e calcular a disparidade.

$$Disp = x_L - x_R \quad (1)$$

Para implementar esse algoritmo, utilizamos a classe StereoSGBM pertencente ao openCV. Essa classe possui um método que computa a disparidade usando uma variação do algoritmo de Hirschmuller. Esse algoritmo utiliza a soma de diferenças absolutas, mas também faz uma análise da região estudada para estimar a real disparidade, com o objetivo de criar suavidade na imagem final, gerando resultados mais confiáveis [5]. Além disso esse método é capaz de computar em escala de subpixel.

A imagem gerada pelo método *compute* da classe SGBM retorna uma imagem em que cada pixel representa a distância do pixel da imagem da direita até a da esquerda, no formato 16bit sinalizados. Esses 16 bits representam um valor em ponto fixo em que os 4 últimos bits são fracionários. Como, para a nossa aplicação, não precisávamos de tanta precisão, ignoramos os bits fracionários e tratamos apenas da parte inteira.

A matriz resultante foi então normalizada para que o range de valores fosse de 0 a 255 seguindo a fórmula:

$$pixel_{NORMALIZADO} = \frac{255 \cdot pixel}{max - min} - \frac{255 \cdot min}{max - min} \quad (2)$$

Para retornar o mapa normalizado para os valores reais em pixels, basta aplicar a equação inversa.

092 2.1.2 Encontrando o mapa de profundidade

093 Em imagens estéreo, objetos mais próximos da imagem possuem uma maior disparidade em
 094 relação aos objetos mais distantes, tornando o mapa de profundidade inversamente propor-
 095 cional ao mapa de profundidade.

096 As coordenadas no mundo podem ser obtidas através das coordenadas dos pixel, con-
 097 hecendo a distância focal das câmeras f e a distância real entre elas b .

$$099 \quad X = \frac{b(x_L + x_r)}{2(x_L - x_r)} \quad Y = \frac{b(y_L + y_r)}{2(x_L - x_r)} \quad Z = \frac{bf}{2(x_L - x_r)} \quad (3)$$

102 O mapa de profundidade também precisou ser normalizado.

104 2.2 Requisito 2

106 Nesse requisito, devemos encontrar o mapa de disparidade e o de profundidade assim como
 107 no requisito 1, contudo as imagens base não estão retificadas. As imagens usadas foram duas
 108 imagens do Morpheus (Figura 1) feitas usando câmeras convergentes. O primeiro passo ao
 109 encontrar imagens convergêntes é retificar. E esse foi o maior desafio que tivemos em todo
 110 o trabalho.



121 Figure 1: Imagens Estéreo (a)MorpheusL (b)MorpheusR não retificadas

123 2.2.1 Tentativas de retificar a imagem

- 125 • Usando a pseudo-inversa da matriz de intrínsecos

126 Para retificar as imagens, tentamos usar a pseudo-inversa da matriz de intrínsecos de
 127 uma das duas câmeras. Como o ponto real é o mesmo para as duas imagens, é possível
 128 isolar as coordenadas homogêneas de uma das duas imagens (equação 4). Contudo,
 129 esse método não forneceu resultados satisfatórios.

$$131 \quad x_L = P_L \cdot (P_R^T P_R)^{-1} \cdot P_R^T \cdot x_R \quad (4)$$

- 134 • Usando a função stereoRectify

135 Pesquisamos na biblioteca OpenCV e encontramos uma função que recebe os parâmet-
 136 ros intrínsecos e extrínsecos das câmeras e retorna a matriz de projeção e rotação de
 137 cada imagem para o sistema de coordenadas da outra.

Para usar essa função, devemos encontrar a rotação e translação relativa entre as câmeras, que calculamos usando a fórmulas: 138
139
140

$$R = R_L^{-1} \cdot R_R \quad e \quad T = R_L^{-1} \cdot (T_R - T_L) \quad (5) \quad 141 \quad 142$$

Passamos as matrizes retornadas pela função stereoRectify para a função initUndistortRectifyMap que calcula os mapas de retificação e retira as distorsões. Os dois mapas retornados são passados para a função remap que calcula a transformação do sistema de coordenadas de uma imagem para a outra. 143
144
145
146

Essa abordagem também não gerou bons resultados. 147
148

- **Marcando as correspondências entre as imagens e calculando a matriz fundamental e linhas epipolares** 149
150

A matriz fundamental de uma imagem é tal que, sabendo as correspondências de pontos na imagem 1 (x, y) e na imagem 2 (x', y'), temos: 151
152
153

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = 0 \quad (6) \quad 154 \quad 155 \quad 156$$

Calculando essa matriz, para uma ponto na imagem 1 é possível encontrar uma linha epipolar na imagem 2 em que estará a correspondência desse pixel. 157
158
159

Encontramos na biblioteca do openCV uma função que gera a matriz fundamental a partir de pontos correspondentes entre as duas imagens. A função findFundamentalMat recebe dois vetores de pontos correspondentes em cada imagem e, usando o método de ransac, calcula a matriz fundamental de uma imagem. 160
161
162
163

Usando a função do openCV stereoRectifyUncalibrated e a matriz fundamental encontrada, calculamos a homografia de uma imagem para a outra. 164
165

Com a imagem retificada utilizamos o requisito 1 para encontrar os mapas de disparidade e profundidade. 166
167
168

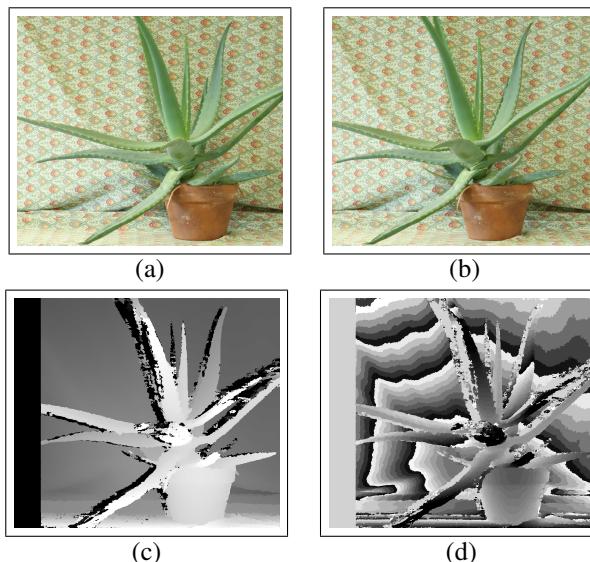
2.3 Requisito 3

Nesse requisito, devemos obter o volume de uma caixa suficientemente grande para armazenar o sofá da imagem do Morpheus. Para isso, usamos as imagens retificadas obtidas no requisito 2. As correspondências dos pontos são feitas usando cliques do mouse nas duas imagens e as coordenadas do mundo são calculadas usando a fórmula descrita em 3. Com as coordenadas do mundo real, os tamanhos das arestas são calculados, apenas calculando a distância entre pontos consecutivos, e o volume é obtido. 171
172
173
174
175
176
177

3 Resultados e Análise

3.1 Requisito 1

Para as imagens aloeR.png e aloeL.png disponibilizadas encontramos o resultado da figura 2. 182
183

184
185
186
187
188
189
190
191200
201 Figure 2: (a)aloeL (b)aloeR (c)Mapa de disparidade (d) Mapa de profundidade
202203 Na imagem de disparidade da planta (Figura 2 (c)), as regiões que possuem cores mais
204 claras são as regiões de maior disparidade e as mais escuras são as de menor disparidade.
205 Apesar de ser a mesma cena, existem regiões em que não há correspondência entre pixels nas
206 duas imagens devido à translação da câmera, essas regiões foram marcadas com preto.207 Na imagem de profundidade (Figura 2 (d)) da planta, o fundo possui um mesmo
208 padrão de cores e formatos, assim, não houve uma correspondência satisfatória dos pixels.
209 Esse fato não foi evidente no mapa de disparidade, mas se apresentou muito marcado no
210 mapa de profundidade. E, como esse mapa também foi normalizado, o ruído se tornou ainda
211 mais evidente.212 Para as imagens babyR.png e babyL.png disponibilizadas encontramos os resultados da
213 figura 3.214 No mapa de disparidade, a imagem *baby* (Figura 3(c)) apresentou mais ruído devido ao
215 padrão repetitivo de cores e geometria do suporte para a boneca, principalmente. Isso difi-
216 cultou a correta correspondência entre os pixels das imagens da direita e da esquerda, oca-
217 sionando o ruído observado.218 O mapa de profundidade também apresentou um padrão inesperado no fundo, mas a
219 parte que representa o bebê está correta.220 Nas imagens de disparidade os pixels que não foram encontrados e os pixels em que a
221 disparidade é zero são marcados como pretos, tornando o mapa ambíguo. Para resolver esse
222 problema, poderíamos marcar os pixels não encontrados com um número negativo e pintá-
223 lo de preto apenas na hora da normalização. Mantendo, na matriz de disparidade real, a
224 informação de que esse ponto não foi encontrado. O mesmo poderia ser feito no mapa de
225 profundidade para os pontos infinitamente longe da imagem e os pontos não encontrados.226

3.2 Requisito 2

227
228 Implementamos as três tentativas de retificação mencionadas em 2.2, a que obteve melhores
229 resultados foi a retificação a partir da matriz fundamental.

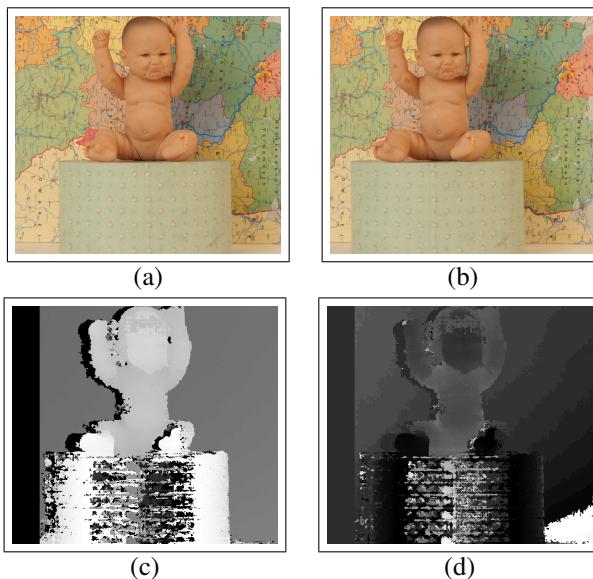


Figure 3: (a) babyL (b) babyR (c) Mapa de disparidade (d) Mapa de profundidade

Marcando 20 correspondências nas imagens, obtivemos a retificação mostrada na figura 4.

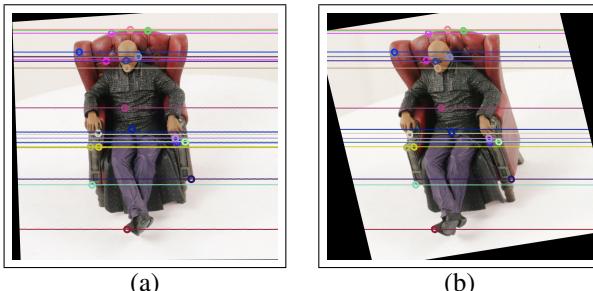


Figure 4: (a) morpheusL retificado (b) morpheusR retificado

A partir dessa retificação encontramos os mapas de disparidade e profundidade, como os do requisito 1.

Os mapas de disparidade e profundidade ficaram ruins, mas é possível distinguir algumas características do morpheus, como as pernas e a cabeça. A qualidade foi prejudicada pela retificação, que não foi ideal, já que os pontos foram marcados com o mouse e estão sujeitos a erro humano. Além disso, a imagem possui muitos pontos de similaridade, que atrapalham o cálculo de disparidade.

3.3 Requisito 3

Como a retificação das imagens do Morpheus não foi ideal, houve ruído e isso causou problemas na hora de calcular os volume. Além disso, as imagens originais, esquerda e direita, não possuíam o mesmo tamanho, o que causou mais distorção. Assim, a disparidade ficou

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

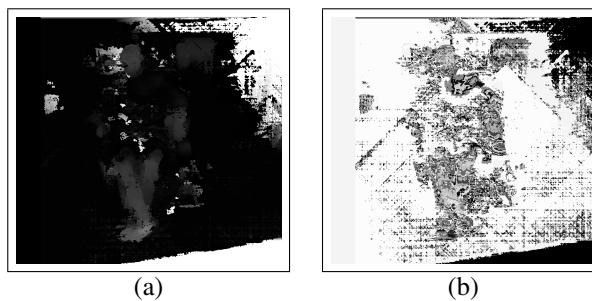
271

272

273

274

275



284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321

Figure 5: (a)Mapa de disparidade (b) Mapa de profundidade

4 Discussão e Conclusões

Com esse projeto, foi possível entender um pouco como funciona uma reconstrução 3D a partir de dados de duas câmeras que capturam a mesma cena. Conseguimos ver que é possível recuperar a informação de profundidade a partir de imagens em duas dimensões. Para essa reconstrução, existem diversos métodos e alguns são mais fáceis de aplicar que outros. O caso mais simples é aquele em que as câmeras estão em paralelo com seus pontos focais alinhados na direção do eixo y. Assim, o método escolhido para tratar imagens com câmeras que não estão nessas condições, foi retificar a imagem para tratá-la no caso mais simples.

Os resultados para imagens estéreo convergentes não foi o ideal, mas ainda foi possível identificar algumas regiões. Porém isso criou um mapa de profundidade muito ruidoso, o que afetou no desempenho do cálculo do volume.

Mesmo não conseguindo bons resultados nos requisitos 2 e 3, avaliamos muito bem esse projeto. Ele exigiu uma extensa pesquisa e várias tentativas que contribuiram bastante para o aprendizado.

References

[1] Gary Bradski Adrian Kaehler.	<i>Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library</i> .	O'Reilly Media, 1 edition, 2016.	322
			323
			324
			325
[2] D.A. Forsyth Adapted by Flávio Vidal.	Visão computacional aula 06 - geometria para múltiplas vistas visão estéreo.	https://aprender.ead.unb.br/course/view.php?id=3183 .	326
			327
			328
			329
[3] Maikon Cismoski dos Santos.	Revisão de conceitos em projeção, homografia, calibração de câmera, geometria epipolar, mapas de profundidade e varredura de planos, 2012.	https://aprender.ead.unb.br/pluginfile.php/574187/mod_resource/content/1/dosSantos_Rocha_Unicamp2012.pdf .	330
			331
			332
			333
[4] Dr. Gerhard Roth.	Homography.	http://people.scs.carleton.ca/~cshu/Courses/comp4900d/notes/homography.pdf .	334
			335
			336
			337
			338
			339
			340
			341
			342
			343
			344
			345
			346
			347
			348
			349
			350
			351
			352
			353
			354
			355
			356
			357
			358
			359
			360
			361
			362
			363
			364
			365
			366
			367