

000  
001 **Projeto Demonstrativo 2**  
002  
003 **Calibração de Câmeras**  
004

005 Carlos Adir Ely Murussi Leite  
006 150121059  
007 carlos.adir.leite@gmail.com  
008 Rebeca Helen Silva de Sousa  
009 130145068  
010 rebeca.hellenss@gmail.com  
011

012 Departamento de Ciência da  
013 Computação  
014 Universidade de Brasília  
015 Campus Darcy Ribeiro, Asa Norte  
016 Brasília-DF, CEP 70910-900, Brazil,  
017

018 **Abstract**  
019  
020

021 A visão computacional precisa frequentemente relacionar o mundo virtual(imagens)  
022 com o mundo real(dimensões físicas). Por isso, desenvolveu-se métodos para calibração  
023 de câmeras, que vinculam essas duas regiões. Esse trabalho tem por objetivo mostrar o  
024 processo de calibração de uma webcam comum de computador, partindo inicialmente  
025 da obtenção dos parâmetros intrínsecos, depois dos extrínsecos e como produto final  
026 realizar a medição de objetos através das imagens obtidas. Devido aos problemas de  
027 calibração encontrados, o trabalho não foi concluído a tempo embora tenha-se repetido  
028 o processo de calibração diversas vezes. Uma das grandes dificuldades encontradas na  
029 elaboração do trabalho consistiu na constante verificação do padrão e do código, pois  
030 não se sabia se os erros observados eram oriundos da captura de fotos ou do algoritmo  
031 que obtém os parâmetros a partir da imagem

032 **1 Introdução**  
033

034 O ramo da visão computacional é bastante diverso e possui variadas aplicações. Para tais  
035 aplicações, frequentemente tem-se o problema de relacionar o cenário digital(as imagens  
036 bidimensionais obtidas por câmeras) com o cenário real  
037 À medida com que uma pessoa cresce, adquire-se a capacidade de relacionar a imagem  
038 obtida pelos olhos com o seu significado real e isso é feito principalmente através de apren-  
039 dizado: aprende-se o tamanho e forma dos objetos do dia a dia e através da inferência,  
040 por comparação pode-se ter uma estimativa de posicionamento de objetos bem como seu  
041 tamanho.

042 Já para câmeras digitais que se comportam de forma diferente ao olho humano na captura  
043 de imagens, existem incertezas relacionadas à câmera que se dividem em dois grupos:

044 **Intrínseco** Inerentes ao processo de fabricação da câmera.  
045

046 **Extrinseco** Relativos à posição da câmera em relação ao espaço.

047 Por esse motivo, utiliza-se conceitos de álgebra linear, geometria e ótica para relacionar o  
048 espaço da imagem obtida pela câmera com a posição real do ponto observado. Essa relação

pode ser dada pela Equação (1) utilizando-se coordenadas homogêneas, em que a matriz  $In$  relaciona os parâmetros intrínsecos e a matriz  $Ex$  relaciona os parâmetros extrínsecos. A Figura (1) mostra alguns dos parâmetros apresentados na Equação (1).

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{In} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}}_{Ex} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (1)$$

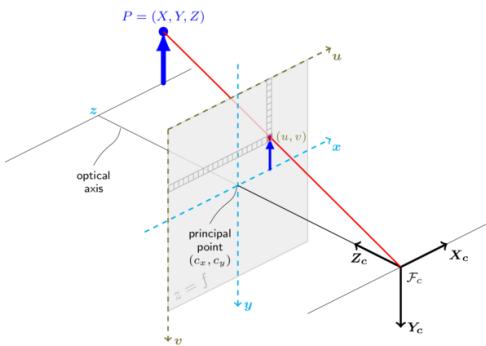


Figure 1: Modelo da câmera Pinhole

Fonte:OpenCV Camera Calibration[1]

É necessário apenas 6 pontos para determinar todos os parâmetros intrínsecos e extrínsecos da câmera. Mas para aumentar a precisão dos parâmetros e diminuir a incerteza associada a cada ponto, pode-se utilizar mais pontos conhecidos e diminuir o erro devido ao ruído. Contudo, como toda a teoria exposta para calibração de câmera considera o modelo de câmera Pinhole e as câmeras digitais são feitas de diversas formas como mostra em Raskar[2]. Devido a essa variação entre câmeras, existem câmeras que autoajustam seu foco e portanto os parâmetros intrínsecos variam.

## 2 Metodologia

A metodologia consistiu em seguir os passos dos requisitos necessários.

Para o primeiro requisito, a metodologia consistiu em adaptar o código implementado em C++ no Projeto Demonstrativo 1.

Para os requisitos posteriores, primeiramente precisou selecionar um tabuleiro de xadrez com  $6 \times 8$  interseções e cada quadrado com 30 mm de lado, imprimiu-se e colou-se em um papelão espesso.

Com o tabuleiro, para o requisito 2, fez 5 pacotes de medições com 25 imagens do tabuleiro em cada pacote. Cada imagem em um pacote são bem similares entre si, enquanto imagem entre pacotes ficam em regiões separadas da imagem para haver uma abrangência maior. Para evitar perda de tempo capturando o padrão frequentemente, planejou-se fazer um código para armazenar as imagens com padrão do tabuleiro assim que fossem capturadas, o qual foi chamado de `get_images`. Uma vez com  $5 \times 25$  imagens do tabuleiro, utilizou-se

092 a função *findChessboardCorners*[¶] para encontrar os 48 pontos para cada imagem. Com o  
 093 conjunto de  $25 \times 48$  pontos em cada pacote, utilizou-se a função *calibrateCamera*[¶] para  
 094 obter no total 5 matrizes e 5 coeficientes de distorção como parâmetros intrínsecos; Com  
 095 todos os parâmetros intrínsecos, pegou-se a média e desvio padrão das 5 matrizes intrínse-  
 096 cas(uma para cada pacote) e dos 5 coeficientes de distorção.

097 Ja os parâmetros extrínsecos também foram obtidos pela função *calibrateCamera*. Como  
 098 tais parâmetros variam conforme a posição do tabuleiro, ou seja, entre os pacotes, não  
 099 podemos pegar a média de varios pacotes como feito com os parâmetros intrínsecos. Para  
 100 isso, pegou-se apenas um pacote de 25 imagens e calculou-se a posição média do padrão à  
 101 câmera. Dos  $25 \times 48$  pontos desse pacote, obteve-se 25 vetores de rotação e 25 vetores de  
 102 translação bem próximos entre si, e depois pegou-se a média dos 25 vetores.

103 Então, para verificar o efeito da distância do padrão sobre os parâmetros extrínsecos, fez-se  
 104 o processo dito acima, variando apenas a distância do tabuleiro à câmera. As 3 distâncias  
 105 escolhidas foram: a mais distante que ainda capturava o padrão, a menor distância que se  
 106 reconhecia todos os pontos do tabuleiro, e uma distância intermediária. Apenas para a dis-  
 107 tância mais próxima era possível apenas 1 pacote de medição, visto que o tabuleiro ocupa a  
 108 tela totalmente. Para cada distância, mediu-se com uma trena a distância entre a câmera.  
 109 Para o requisito 4, foi mesclar os resultados obtidos nos requisitos 2 e 3 com o requisito 1  
 110 pois a interface, os parâmetros intrínsecos e extrínsecos foram obtidos nos requisitos anteri-  
 111 ores.

112 Para implementação do código, seguiu-se a estratégia de utilizar funções já implementadas  
 113 pela OpenCV para encontrar vertices do tabuleiro de xadrez, calcular os parâmetros das  
 114 matrizes *In* e *Ex* da Equação (1). Para implementação primeiramente pensou-se em uti-  
 115 lizar C++, mas por dificuldade na utilização de funções específicas da OpenCV optou-se  
 116 por utilizar Python e utilizou-se um tutorial do OpenCV na utilização das funções[¶]. Para  
 117 escolha da câmera, optou-se por uma webcam, visto que outra câmera que era planejada de  
 118 ser utilizada auto ajustava o seu foco e isso mudava os parâmetros intrínsecos frequente-  
 119 mente.

120

121

### 3 Resultados

122

123 Os requisitos foram separados e divididos pelos requisitos através dos códigos *r1* a *r4*. Para  
 124 o código *r1*, obteve-se simplesmente uma interface do usuário em que com dois cliques do  
 125 mouse era possível medir a distância em píxeis entre dois pontos.

126 Com a captura das imagens pelo algoritmo *get\_images*, obteve-se a média e desvio padrão  
 127 das 5 matrizes intrínsecas e coefientes de distorção através do algoritmo *r2*. Os valores dos  
 128 parâmetros das matrizes intrínsecas são mostrados pela Tabela (3).

129

130 Table 1: Parâmetros intrínsecos da matriz *In* para diferentes posições a uma distância de 75  
 131 cm do plano da câmera

	<i>In</i> <sub>1</sub>	<i>In</i> <sub>2</sub>	<i>In</i> <sub>3</sub>	<i>In</i> <sub>4</sub>	<i>In</i> <sub>5</sub>	<i>In</i> ( $\bar{x} \pm \sigma$ )
<i>f</i> <sub>x</sub>	721	1154	1739	1052	1955	$1324 \pm 455$
<i>f</i> <sub>y</sub>	728	1164	1846	1079	3219	$1607 \pm 884$
<i>c</i> <sub>x</sub>	634	714	639	649	633	$654 \pm 30$
<i>c</i> <sub>y</sub>	374	316	369	334	353	$349 \pm 22$

137

Figure 2: Posições das fotos tiradas do tabuleiro



Uma vez que tenha-se os parâmetros intrínsecos obtidos, fazendo os procedimentos descritos na metodologia pôde-se estimar os valores dos parâmetros extrínsecos. Utilizando os vetores fornecidos pela função *calibrateCamera*, e calculado a média com os 25 valores para cada uma das 3 distâncias, obtemos a Tabela (3). Os valores das incertezas dos vetores foram todos menores que 5%.

Table 2: Parâmetros extrínsecos da matriz *Ex* para diferentes distâncias no eixo focal

	Vetor Rotação			Vetor Translação			
	35 cm	-0.190	0.06	0.001	-2.61	-2.33	11.78
75 cm	-0.31	0.23	-0.016	2.89	0.70	33.40	
120 cm	-0.45	0.017	-0.021	-1.10	10.89	75.5	

Uma vez obtidos os vetores médios, precisou-se apenas da matriz *R* responsável pela rotação. Para obtenção dela a partir do vetor de rotação, utilizou-se a função *Rodrigues* da OpenCV. Com a matriz *R* e o vetor de translação, bastou montar a matriz que transforma as coordenadas do mundo tridimensional(em coordenadas homogêneas) para as coordenadas em pixels segundo a Equação (1). Para a distância de 75 cm, tem-se a matriz extrínseca mostrada na Equação (2)

$$[R|t] = \begin{bmatrix} -0.9733 & -0.0195 & 0.2286 & 2.8870 \\ -0.0517 & 0.9521 & 0.3015 & 0.7016 \\ -0.2236 & -0.3053 & 0.9257 & 33.405 \end{bmatrix} \quad (2)$$

Após obtenção da matriz extrinseca, tentou-se aplicar para cumprir o requisito 4. Mas das 15 tentativas de calibração realizadas, a imagem ficava mais distorcida que o normal, ou seja, mais distorcida que não aplicar qualquer correção. As grandes deformações nas laterais apareciam durante a calibração dos valores intrínsecos indicando um erro na calibração dos valores intrínsecos. E a movimentação para baixo do centro da imagem, como mostra a Figura (3), indica que há um erro também na calibração dos valores extrínsecos.

## 4 Discussão e Conclusões

Podemos ver que faltou dois requisitos a serem cumpridos, que é realizar a medição do objeto utilizando a calibração. Contudo, pôde-se aprender e perceber diversos problemas que ocorrem na calibração e algumas formas de correção.

138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183

184  
185  
186  
187  
188  
189  
190  
191  
192



193      Figure 3: Produto final após a correção dos valores intrínsecos e extrínsecos  
194

195  
196      Uma tática de calibração para ser implementada que poderia corrigir o problema é tratar  
197      cada região da imagem independentemente e assim cada região ter seu padrão de calibração  
198      reduz a distorção observada. Uma desvantagem desse método é a dificuldade de calibração  
199      além do custo computacional que eleva-se.

200  
201 

## References

202

- 203 [1] OpenCV Community. Camera calibration and 3d reconstruction, 2018. URL  
204      [https://docs.opencv.org/2.4/modules/calib3d/doc/camera\\_](https://docs.opencv.org/2.4/modules/calib3d/doc/camera_)  
205      [calibration\\_and\\_3d\\_reconstruction.html](calibration_and_3d_reconstruction.html).
- 206 [2] OpenCV Community. Camera calibration tutorial, 2018. URL [https://docs.opencv.org/3.1.0/dc/dbb/tutorial\\_py\\_calibration.html](https://docs.opencv.org/3.1.0/dc/dbb/tutorial_py_calibration.html).
- 207 [3] Ramesh Raskar. *Computational Photography*. Cambridge, MA 02139, USA, 2007.  
208      URL <http://web.media.mit.edu/raskar/>.

209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229