# Projeto Demonstrativo 2

Raphael Soares 14/0160299
raphael.soares.1996@gmail.com

Departamento de Ciência da Computação
Universidade de Brasília
Campus Darcy Ribeiro, Asa Norte
Brasília-DF, CEP 70910-900, Brazil,

**Abstract**

Este segundo projeto tem como objetivo principal a avaliação dos aspectos envolvidos em calibração de câmeras. Para isso, foi desenvolvida uma "régua visual" que tenta estimar a altura ou largura de um objeto através apenas da sua imagem capturada pela câmera. Ou seja, através da medida em pixels de um objeto, o programa é capaz de estimar a medida real, em centímetros, deste objeto. Para atingir esse objetivo foi necessário transformar coordenadas em pixels da imagem em coordenadas tridimensionais do mundo real usando os parâmetros extrínsecos e a matriz de parâmetros intrínsecos da câmera, que por sua vez foi obtida através da calibração da câmera realizada.

## 1 Introdução

A visão começa com a detecção de luz do mundo. Esta luz começa com raios emanando de alguma origem, que viajam pelo espaço até atingir algum objeto. Quando esta luz atinge algum objeto, muito dela é absorvida, e o que não é absorvido nós percebemos como a cor do objeto. A luz refletida que encontra o caminho até o nosso olho (ou nossa câmera) é colecionada na nossa retina/imager (ou nosso "filme/aparelho"). A geometria desse arranjo (particularmente da viagem dos raios do objeto através da lente em nossos olhos ou câmera e para a retina ou filme) é de grande importância para a prática de visão computacional. Um modelo simples, porém útil de como isso acontece é o modelo da câmera pinhole apresentado em [2]. Infelizmente, um pinhole real não é uma boa forma de obter imagens porque ele não obtém luz suficiente para exposições rápidas e isso é um dos motivos de nossas câmeras e olhos usarem lentes, porque dessa forma podemos obter mais luz do que estaria disponível em um ponto.

Este projeto tem como objetivo usar calibração de câmeras para corrigir (matematicamente) desvios principais que o uso de lentes nos traz, as distorções. Existem dois tipos principais de distorções de lentes: radial, que é resultado da forma das lentes; e a tangencial, que surge do processo de montagem da câmera. Na teoria é possível definir lentes que não introduzem distorções, porém na prática nenhuma lente é perfeita. A calibração de câmeras é importante também, para relacionar as medidas da câmera com as medidas no mundo real tridimensional. Essa relação é importante não só porque as cenas são tridimensionais, mas porque elas também são espaços físicos com unidades físicas. Portanto, a relação entre a unidade natural da câmera (pixels) e as unidades do mundo físico (metros, por exemplo) é um componente crítico para reconstruir uma cena tridimensional ou construir uma régua visual.

# 2 Metodologia

Nesta seção são apresentados os métodos e procedimentos utilizados em cada um dos requisitos para obter os resultados pedidos.

## 2.1 Requisito 1

Assim como no projeto demonstrativo 1, foram usados dois booleanos $x$ e $y$ para verificar se o primeiro e o segundo clique do mouse foram dados, respectivamente. Também há um terceiro booleano $z$ que indica se houve ou não clique no frame em questão. O algoritmo funciona da seguinte forma:

- Ao dar um clique do mouse, primeiramente é verificado se $x$ é verdadeiro. Em caso afirmativo, $y$ é setado para verdadeiro e desenhamos a linha usando os pontos armazenados anteriormente.
- Em caso negativo, setamos $y$ para falso.
- Trocamos o valor verdade atual de $x$ caso um clique seja dado. Isso se faz necessário para o caso de haver mais de dois cliques do mouse.
- Se houve clique então $z$ é setado para verdadeiro, caso contrário é setado para falso. O booleano $z$ é usado para desenharmos a linha nos frames seguintes mesmo que não haja clique, caso seja necessário. Para isso o passo 2 é essencial, pois precisamos saber se houveram dois cliques ou não.

O algoritmo "apaga" uma linha já desenhada caso um novo clique do mouse seja dado e já considera este novo clique como um primeiro clique para uma nova linha. Além disso, as coordenadas dos pixels são salvas conforme os cliques são dados. Essas coordenadas são necessárias não só para que a linha seja desenhadas nos frames seguintes do vídeo, mas também para que o cálculo da estimativa da altura ou largura do objeto seja feito no 2.4.

## 2.2 Requisito 2

O OpenCV [4] fornece vários algoritmos para nos ajudar a computar os parâmetros intrínsecos e o vetor de distorção. A calibração é feita via *cv::calibrateCamera()*, que já fornece os vetores de translação, rotação, o vetor de distorção e a matriz com os parâmetros intrínsecos da câmera. Para cada imagem que a câmera captura de um objeto particular, nós podemos descrever a pose do objeto relativo ao sistema de coordenadas da câmera em termos da rotação e da translação. O vetor de translação é como nós representamos um deslocamento de um sistema de coordenadas para outro cuja origem é deslocada para outra localização. Ou seja, o vetor de translação é apenas o deslocamento da origem do primeiro sistema de coordenadas para o segundo. Assim, para mudar de um sistema de coordenadas centrado em um objeto para um centrado na câmera, o vetor de translação apropriado é: $\overrightarrow{T} = origin_{object} - origin_{camera}$. Dessa forma, é possível notar pela **Figura** 1 que um ponto das coordenadas do objeto (ou mundo) $\overrightarrow{P_o}$ tem coordenadas $\overrightarrow{P_c}$ nas coordenadas do frame da câmera: $\overrightarrow{P_c} = R \cdot (\overrightarrow{P_o} - \overrightarrow{T})$. Esta equação combinada com as correções intrínsecas da câmera irão formar o sistema de equações que o OpenCV irá resolver. A solução para estas equações contém os parâmetros de calibração da câmera.

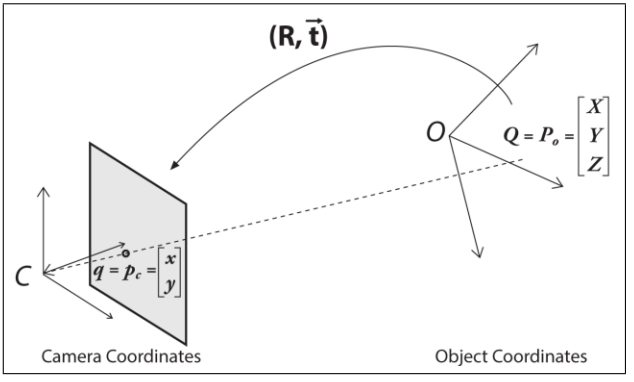


Figure 1: Convertendo do sistema de coordenadas do objeto para o sistema de coordenadas da câmera: o ponto P no objeto é visto como o ponto p no plano da imagem; nós relacionamos o ponto p com o ponto P aplicando uma matriz de rotação R e um vetor de translação t em P. Fonte: [5]

No OpenCV [5] nós temos 4 parâmetros associados a matriz intrínseca da câmera, cinco (ou mais) parâmetros de distorção, que consistem de três (ou mais) parâmetros radiais, e dois tangenciais. Os parâmetros intrínsecos controlam a transformação linear de projeção que relacionam o objeto físico com a imagem produzida. Na teoria seria necessário apenas três pontos de cantos em um padrão conhecido para resolver os nossos 5 parâmetros de distorção. Logo, apenas uma *"screenshot"* do tabuleiro de xadrez seria suficiente. Entretanto, devido ao casamento dos parâmetros intrínsecos com os extrínsecos, apenas uma *"screenshot"* não é suficiente. Pode-se notar que os parâmetros extrínsecos incluem três parâmetros de translação e três de rotação dando um total de 6 por imagem do tabuleiro de xadrez. Junto com os 4 parâmetros da matriz dos intrínsecos da câmera nós temos um total de 10 que precisamos resolver, em um caso de uma única imagem, e 6 adicionais para cada imagem. Supondo que temos $N$ cantos e $K$ imagens do tabuleiro de xadrez (em posições diferentes) é necessário ter $2 \cdot N \cdot K \geq 6 \cdot K \cdot 4((N-3) \cdot K \geq 2)$, onde $K \geq 1$ para resolver o problema de calibração. Entretanto, conforme apresentado em [5], para resultados de alta qualidades é necessário pelo menos 10 imagens de um xadrez 7 x 8 ou maior, na prática. Esta disparidade entre as 2 imagens na teoria, e 10 ou mais requeridas na prática, é resultado de um alto grau de sensibilidade que os parâmetros intrínsecos possuem, mesmo com pouco ruído. Por isso, nesse passo, em vez de fazer a média dos parâmetros de calibração obtidos de algumas *snapshots* do tabuleiro, foram obtidas **24 snapshots/screenshots** para obter a matriz de calibração, vetores de translação, rotação e coeficientes de distorção. Sendo que os 8 primeiros *snapshots* foram obtidos a uma distância de 30 cm do centro da câmera ($d_{min}$), os 8 seguintes a uma distância de 44 cm ($d_{med}$) e os 8 últimos a uma distância de 58 cm ($d_{max}$). Depois, foi calculada a média e desvio padrão para cada um dos três grupos de 8 vetores de translação obtidos com esses 24 *snapshots*.

## 2.3 Requisito 3

O parâmetro *objectPoints* da função *cv::calibrateCamera()* foi definido da seguinte forma no programa: o primeiro canto no tabuleiro de xadrez está no ponto (0,0,0), o próximo

no (0,1,0), o próximo (0,2,0), e assim em diante. Dessa forma, a escala do vetor *tvec* de translação da saída da função *cv::calibrateCamera* foi implicitamente alterada. Ou seja, ao definir os pontos dos cantos dos tabuleiros de xadrez desta forma, as distâncias são medidas em "quadrados do tabuleiro"[1]. Logo, ao calcular a norma do vetor de translação foi necessário depois multiplicar pela largura ou altura do quadrado do tabuleiro (na unidade correspondente das distâncias) para comparar com as distâncias $d_{min}, d_{med}, d_{max}$.

Os parâmetros extrínsecos são os vetores de rotação e translação (*tvec e rvec*), que transformam pontos definidos no *frame* das coordenadas do mundo para pontos definidos no *frame* da camera. Eles são saídas da função *cv::calibrateCamera()* e são apresentados no terminal durante a execução do programa. Cada vetor possui 3 elementos, correspondentes aos eixos *x, y, z*, e temos *n* de cada um desses dois vetores, onde *n* é o número de capturas do tabuleiro.

## 2.4 Requisito 4

Para a imagem sem distorção, primeiro foi computado os "*undistortion maps*" e depois foram aplicados na imagem através da função *cv::remap()*. O motivo para esta separação, em vez de usar a função *cv::undistort()*, é porque assim só é necessário calcular os *undistortion maps* uma única vez usando os parâmetros de calibração, e depois apenas aplicá-los na imagem, conforme novos frames do vídeo vão chegando.

Para estimar a largura/altura do objeto é necessário mapear um ponto 2D no plano da imagem para um ponto no espaço 3D. Este processo é conhecido como *back-projection*, que consiste em projetar um ponto $q$ da imagem no conjunto de pontos do espaço 3D, sendo que os pontos 3D pertencentes a este conjunto constituem um raio no espaço, o qual passa pelo centro de projeção da câmera. Conforme apresentado por [7], para [6] é possível calcular o raio $Q(\lambda)$ que passa pelo centro da câmera para obter as coordenadas do mundo real usando, entre outros fatores, a matriz de rotação, a matriz com os parâmetros intrínsecos da câmera e o ponto no plano da imagem.

Contudo, como a distância $d \in \{d_{min}, d_{med}, d_{max}\} = \{30, 44, 55\}$ entre o objeto a se medir e o centro da câmera é conhecida (profundidade), e o objetivo é apenas de estimar medidas, foi usado um método mais simples de estimação das medidas do objeto [8]. A coordenada $X, Y, Z$ do mundo real foi obtida usando a seguinte equação:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = dK^{-1}q \tag{1}$$

onde $q = (x, y, 1)^T$ é o ponto no plano da imagem. Aqui, a profundidade do objeto utilizada foi considerada nas distâncias $d$ pedidas. Ou seja, na teoria a parte frontal do objeto, que possui forma exata de um paralelepípedo com largura e profundidade de 2,5 cm, está a uma distância de 27,5 cm do centro da câmera para $d_{min}$, por exemplo.

Para estimar a altura/largura de um objeto em uma imagem basta obter dois pontos no plano da imagem e calcular a distância euclidiana desses dois pontos mapeados no mundo real, independente do método utilizado. Um outro método que pode ser utilizado para estimação da pose da câmera é o apresentado por [1], que consiste em usar a matriz de projeção para obter os parâmetros internos e externos de calibração. Além disso, usando um fator de

[1] Em contrapartida, os parâmetros da matriz intrínseca da câmera são sempre reportados em pixels


Figure 2: Linha sendo desenhada na janela raw após dois cliques do mouse.

escala é possível obter as coordenadas do mundo real correspondentes a um ponto na imagem através da matriz de projeção. Também é possível obter a posição da câmera no frame do mundo usando a transformação euclidiana inversa.

# 3 Resultados

Nesta seção são apresentados em forma de figuras e tabelas os resultados da aplicação para cada um dos requisitos.

## 3.1 Requisito 1

Apesar de não estar sendo mostrado na **Figura** 2 o comprimento é mostrado no terminal de execução, após a linha ser desenhada. Também é mostrado os dois pixels clicados.

## 3.2 Requisito 2

A matriz intrínseca da câmera talvez é o resultado final mais interessante da calibração de câmeras, porque ela nos permite transformar coordenadas tridimensionais para coordenadas bidimensionais na imagem. Na **Figura** 3 podemos ver a janela da imagem original e a sem distorção, após o processo de calibração ter sido realizado. Os resultados provenientes da calibração são apresentados no terminal do programa. Com cliques do mouse as linhas podem ser desenhadas nas duas janelas.

## 3.3 Requisitos 3 e 4

Os dados pedidos no requisito 3 e 4 são apresentados na **Tabela** 3.3. $P_1$ e $P_2$ representam as coordenadas $x$ e $y$ do primeiro e segundo pixel correspondentes ao primeiro e segundo

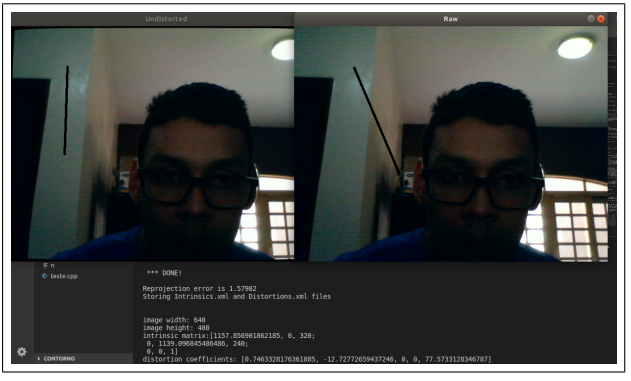


Figure 3: Linhas sendo desenhadas nas janelas raw e undistorted após dois cliques do mouse em cada uma.

clique do mouse dados para estimar a altura ou largura do objeto na imagem. Para $d_{min}$ a norma do vetor de translação pela calibração extrínseca é a norma dos 8 primeiros vetores de translação do total de 24, e assim em diante para as outras distâncias, conforme explicado anteriormente em 2.2.

Table 1: Tabela contendo os dados pedidos nos Requisitos 3 e 4. As medidas são fornecidas em centímetros.

| Posição | $d_{min}$ (cm) | $d_{med}$ (cm) | $d_{max}$ (cm) |
|---|---|---|---|
| $\|\|t\|\|$ da trena | 30 | 44 | 58 |
| $\|\|t\|\|$ da calibração | Média = 35.84394<br>$\sigma = 19.73154$ | Média = 46.44475<br>$\sigma = 24.78756$ | Média = 60.49741<br>$\sigma = 32.18481$ |
| $I_{raw,centre}$ | 2.330139<br>$P_1$ = (296,249)<br>$P_2$ = (359,254) | 2.487273<br>$P_1$ = (264,245)<br>$P_2$ = (310,245) | 2.356549<br>$P_1$ = (315,286)<br>$P_2$ = (348,288) |
| $I_{raw,perifery}$ | 2.543802<br>$P_1$ = (17,292)<br>$P_2$ = (86,292) | 2.604700<br>$P_1$ = (50,298)<br>$P_2$ = (98,302) | 2.712342<br>$P_1$ = (280,373)<br>$P_2$ = (330,373) |
| $I_{undistorted,centre}$ | 2.433202<br>$P_1$ = (242,222)<br>$P_2$ = (308,222) | 2.379765<br>$P_1$ = (277,254)<br>$P_2$ = (321,255) | 2.494646<br>$P_1$ = (288,299)<br>$P_2$ = (323,299) |
| $I_{undistorted,perifery}$ | 2.618607<br>$P_1$ = (20,278)<br>$P_2$ = (91,280) | 2.541938<br>$P_1$ = (133,286)<br>$P_2$ = (180,287) | 2.498846<br>$P_1$ = (173,278)<br>$P_2$ = (208,280) |

# 4 Discussões e Conclusões

Para distorções radiais, a distorção é 0 no centro óptico do aparelho e cresce assim que movemos para a periferia. Na prática, esta distorção é pequena e pode ser caracterizada por poucos termos de uma expansão da série de Taylor em torno de $r = 0$. Isso explica porque as

medidas do objeto tiradas próximo ao centro da imagem se aproximam mais da medida real $l$, onde $l = 2.5$ cm . Além disso, algumas medidas da largura poderiam ser mais precisas se uma linha exatamente reta pudesse ser desenhada. Por exemplo, em $I_{raw,centre}$ para $d_{min}$ a coordenada $y$, que corresponde a altura, não deveria ser alterada. Entretanto, podemos observar que as medidas estimadas foram bastante próximas da medida (largura l) real. Os resultados foram satisfatórios.

# References


[1] Teófilo Emídio de Campos. *3D Visual Tracking of Articulated Objects and Hands*. PhD thesis, University of Oxford, 2006.

[2] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN 0521540518, 2003.

[3] *The OpenCV Reference Manual*. Itseez, 3.2.0 edition, Abril 2014.

[4] Itseez. Open source computer vision library. https://github.com/itseez/opencv, 2015.

[5] Adrian Kaehler and Gary Bradski. *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*. O'Reilly Media, Inc., ISBN 978-1-4919-3800-3, 2016.

[6] Yannick Morvan. *Acquisition, compression and rendering of depth and texture for multi-view video*. PhD thesis, Eindhoven University of Technology, Abril 2009.

[7] *Revisão de Conceitos em Projeção, Homografia, Calibração de Câmera, Geometria Epipolar, Mapas de Profundidade e Varredura de Planos*. Unicamp, Junho 2012.

[8] Stackoverflow users. Measure real size of object with calibrated camera opencv c++? https://goo.gl/7rZwM8, 2018.