Projeto Demonstrativo 2 - Calibração de Câmeras (15/04/2018)

Tiago de Carvalho Gallo Pereira (14/0164138)
Universidade de Brasília - UnB
Princípios da visão computacional - Turma A
tiagodecarvalhopereira@gmail.com

Abstract

Projeto demonstrativo para aprender a calibrar uma câmera digital, de forma a eliminar as distorções radiais e tangenciais que podem conter nela. Com a câmera calibrada, utilizar métodos para calcular a matriz de valores extrínsecos da câmera e utilizar isso para fazer uma transformação dos pontos na imagem para pontos no mundo 3D, considerando Z = 0 (eliminando o fato profundidade).

1. Introdução

A calibração de câmeras digitais com os cálculos dos valores extrínsecos e intrínsecos da mesma, é o primeiro passo para se fazer uma aplicação que utilize câmeras estéreo o múltiplas câmeras. São chamadas de câmeras estéreo o conjunto de duas câmeras próximas e apontadas na mesma direção, elas simulam a vista humana. Já múltiplas câmeras podem estar organizadas de qualquer maneira, apontadas para vários pontos de uma sala ou para o mesmo ponto, mas de diferentes ângulos.

A utilização dessas geometrias de câmeras abre campo para várias aplicações interessantes. As câmeras estéreo podem ser usadas para gravar filmes que futuramente serão reproduzidos em 3D, o uso de duas câmeras facilita a preparação das imagens para esse tipo de filme. Já a geometria com câmeras múltiplas é muito utilizada no cinema, para efeitos especiais, pois para preparar um efeito especial pode ser necessário conhecer os tamanhos de cada pedaço da cena e não seria viável uma pessoa medir tudo a cada cena, portanto com uma estimação do mundo 3D a partir das imagens de múltiplas câmeras podemos ter um resultado muito bom para preparar efeitos especiais.

2. Metodologia

2.1. Requisito 1

O requisito 1 desse projeto é muito simples, para alcançar os resultados desejados foi mostrada a imagem da webcam em uma janela e permitiu-se que o usuário clicasse em dois pontos aleatórios da imagem. Quando o usuário clica no segundo ponto, o programa pega os dois pontos e calcula a distância entre eles usando distância euclidiana (equação 1).

$$dist = \sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2} \tag{1}$$

2.2. Requisito 2

O objetivo do requisito 2 era calcular a matriz de parâmetros intrínsecos da câmera, para isso foi utilizado, como recomendado pelo professor, o tutorial do OpenCV que ajuda a fazer esses cálculos ([3]).

A matriz dos parâmetros intrínsecos da câmera mostrada na equação 2 e os coeficientes de distorção da imagem são utilizados para corrigir a distorção gerada pela câmera, para isso mapeamos os pontos (x,y) da imagem de acordo com a equação 3.

$$K = \frac{1}{f}\begin{pmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2}$$

$$x = x_d(1 + k_1 r^2 + k_2 r^2) \tag{3}$$

Após calcular os valores da matriz de parâmetros intrínsecos e de distorção, salvamos esses valores em um arquivo XML dentro de uma pasta chamada XMLs com o nome "calibration1.xml", na vez seguinte em que o programa é executado é gerado um arquivo com o nome "calibration2.xml"e assim por diante. Sempre que o programa é executado são gerados dois arquivos, "distortion.xml"e "intrinsic.xml"contendo a média e o desvio padrão dos valores da matriz de parâmetros intrínsecos e os coeficiente de distorção, esses dois arquivos serão utilizados nos próximos requisitos para pegar esses valores já calculados.

Para calcular ambos os valores foram capturadas 10 imagens onde o programa conseguia encontrar os pontos do padrão de trabuleiro de xadrez, com um intervalo de 1s entre cada imagem. Além do calculo desses valores, na execução

desse programa, ainda mostramos duas janelas, uma contendo a imagem pura da webcam e outra contendo a imagem com a distorção corrigida.

2.3. Requisito 3

No requisito 3 foi utilizado parte do código do requisito 1 para pegar 9 imagens onde o programa consegue detectar o padrão do tabuleiro de xadrez, onde 3 dessas imagens estão a uma distância muito próxima da câmera, 3 delas estão a uma distância média da câmera e as outras 3 estão a uma distância longe da câmera. Todas essas imagens contém um objeto ao lado do tabuleiro de xadrez (no mesmo plano Z = 0) e elas foram salvas na pasta "images", pois elas serão utilizadas posteriormente pelo programa do requisito 4 para calcularmos as dimensões desse objeto que está ao lado do tabuleiro de xadrez.

Nesse Requisito foi pedido também o calculo da matriz dos parâmetros extrínsecos da câmera, para isso podemos pegar os pontos encontrados no tabuleiro de xadrez e associarmos eles com pontos do mundo 3D que nós conhecemos, assumindo que a origem do mundo é na primeira interseção de quatro quadrados do jogo de xadrez e que o eixo x aponta na direção do crescimento das colunas da imagem e o eixo y na direção de crescimento das linhas da imagem. Ao associar todos esses pontos por meio das equações 4 e **??** encontrada na literatura [1], obtivemos um sitema linear homogêneo, ou seja do formato da equação **??**.

$$x_i = \frac{p_{11}X_i + p_{12}Y_i + p_{13}Z_i + p_{14}}{p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p_{34}} \tag{4}$$

$$y_i = \frac{p_{21}X_i + p_{22}Y_i + p_{23}Z_i + p_{24}}{p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p_{34}} \tag{5}$$

$$Ap = 0 \tag{6}$$

O método mais simples de resolver esse sistema é dizer que p = 0, no entanto essa não é uma solução válida para nós, pois não vamos conseguir nada com uma matriz de zeros. Logo, devemos partir para uma abordagem mais algébrica, muitas literaturas recomendam utilizar métodos como SVD (single value decomposition) ou eigenvalues e eigenvectors, no entanto eu não consegui implementar esses métodos corretamente. Contudo utilizei a função SolvePnP do OpenCV que encontrei no tutorial desse blog [2]. Essa função já me retorna o vetor de translação e o vetor de rotação, me poupando ainda do trabalho de ter que usar QR decomposition na matriz p para encontrar os vetores de translação e rotação.

2.4. Requisito 4

Para o requisito 4 foi necessário apenas juntar todos os requisitos anteriores, utilizando as matrizes de parâmetros

```
Primeiro clique na coordenada (253,105)
Segundo clique na coordenada (330,119)
A distancia euclidiana dos pontos foi 78.26237921249263
```

Figura 1. Cálculo da distância entre dois pontos gerado pelo programa do requisito 1

intrínsecos e a matriz de parâmetros extrínsecos conseguimos gerar a matriz de projeção P de acordo com a equação **??**.

$$P = \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \tag{7}$$

Após obtermos a matriz de projeção P, bastou utilizar o requisito 1 para escolher dois pontos na imagem e mapear esses dois pontos para o mundo por meio da equação 8

$$P^{-1} \begin{bmatrix} \frac{x}{s} \\ \frac{y}{s} \\ s \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{8}$$

Depois de termos os pontos no mundo, utilizamos a equação 1 para calcular o tamanho real do objeto.

3. Materiais e métodos

Para a realização desse experimento foi utilizado um ambiente instalado com Python 3.5.2, OpenCV 3.2.0 e NumPy 1.14.2. Para cada um dos requisitos foi criado um arquivo com o código fonte em python, em cada arquivo foi criada uma classe que cumpria os requisitos desejados.

Os materiais externos usados para esse projeto demonstrativo foram, uma régua milimetrada com precisão de 0.05mm, o padrão de calibração (tabuleiro de xadrez com quadrados 29mmx29mm) colado em um livro para dar suporte, um celular de 14.3cm (tamanho medido do conjunto celular + capinha) para testar o funcionamento do programa e uma webcam Logitech c270.

4. Resultados

4.1. Requisito 1

O resultado desse requisito é muito simples, após clicar em dois pontos na imagem temos a saída no terminal mostrada na imagem 1

4.2. Requisito 2

Esse programa gerou arquivos xml com os valores médios e de desvio padrão de vários coeficientes que desejou-se calcular, um exemplo de arquivo de saída desse programa pode ser visto na imagem 2

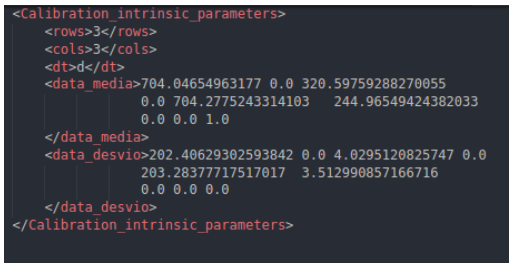

Figura 2. Arquivo xml contendo os valores médios e desvio padrão da matriz de coeficiente intrínsecos gerado pelo requisito 2

Tabela 1. Distâncias estimadas pela norma do vetor de translação

	distância real	distância estimada	desvio padrão
perto	54cm	52,8cm	0,129cm
médio	140cm	130,3cm	0,284cm
longe	243cm	212,2cm	8,253cm

4.3. Requisito 3

O requisito 3 gera dois dados de saída um deles é a norma média do vetor de translação (que estima a distância entre o plano do tabuleiro de xadrez e a câmera) que pode ser verificada na tabela 4.3 e a outra é um arquivo xml contendo informações sobre o vetor de translação e a matriz de rotação que pode ser verificado na imagem 3

4.4. Requisito 4

O requisito 4 quando utilizado em uma das imagens salvas pelo requisito 3, conseguiu mostrar na saída a imagem 4. No entanto a tabela 4.4 que pôde ser gerada com análise nas saídas do requisito 4 é a informação que mais nos agrega valor. O objeto medido em todos os casos era um celular com medidas reais de 14,3cm

Figura 3. Arquivo xml contendo os valores da matriz de rotação e vetor de translação gerado pelo requisito 3

Tabela 2. Medidas usando a régua virtual para o mesmo objeto

Posição do padrão de calibração	d_{min}	d_{med}	d_{max}
$l_{raw,centre}$	14,1cm	14,0cm	13,7cm
$l_{raw,perifery}$	13,6cm	13,8cm	13,6cm
$l_{undistorted,centre}$	13,8cm	13,9cm	13,7cm
$l_{undistorted,perifery}$	13,6cm	13,7cm	13,8cm

5. Discussão e Conclusões

Pela tabela 4.3 podemos ver que o vetor de translação aumenta o erro conforme a distância entre o padrão do tabuleiro de xadrez e a câmera aumenta. Isso provavelmente ocorre, pois fica mais difícil para o programa identificar corretamente os pontos exatos do tabuleiro de xadrez, além do mais há menos pixels para mapear o tabuleiro do jogo de xadrez, então a sensibilidade é reduzida. Como o pixel é um valor inteiro, a precisão do mapeamento do ponto quando o padrão fica muito longe da tela é menor e com isso a relação entre tamanho no mundo real e na imagem diverge bastante.

No entanto isso não é um grande problema para a nossa régua virtual como pode ser verificado na tabela 4.4, isso ocorre pois a maior influência do vetor de translação está na coordenada Z, mas como estamos considerando que a origem do mundo, ponto (0,0,0), está no tabuleiro de xadrez e que o tabuleiro de xadrez é paralelo ao plano formado pelos eixos XY (ou seja, Z = 0 para todos os pontos do tabuleiro), então a coordenada Z do vetor de translação não é considerada na nossa matriz de projeção e com isso os resultados são melhores que o esperado mesmo para objetos distantes da câmera.

Obviamente, o fator principal para melhorar os resultados seria trocar a webcam por uma câmera melhor. No entanto não podemos sempre querer melhorar os nossos resultados apenas melhorando o hardware, pois há um limite para essa melhoria e nem sempre temos condição de conseguir tal melhoria. Portanto, alguns fatores que poderiam melhorar os resultados sem ter que melhorar o hardware são, ter alguma referência para conseguir deixar o tabuleiro de xadrez o mais paralelo possível ao plano formado pelos eixos XY, fazer o trabalho em dupla pois em várias ocasiões é necessário apertar um botão no teclado e ao mesmo tempo estar em uma distância razoável para que a webcam te enxergue com o padrão ou ter que segurar o tabuleiro de xadrez a uma certa distância da câmera e medir essa distância

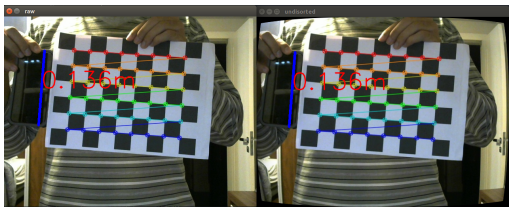

Figura 4. Medida da régua virtual implementada no requisito 4

com a trena, esses tipos de operação são muito complicadas de serem feitas sozinho e acarreta em erros nos resultados.

Referências

[1] `Aulas do professor Vidal, disponibilizadas no moodle`

[2] Learnopencv. *Head pose estimation using opencv.* Blog post link. Acessado em 14 de abril de 2018.

[3] OpenCV. *Camera Calibration.* Tutorial link. Acessado em 13 de abril de 2018.