Visão por Computador 2016-17, Guia Prático N.º 5

Rui Oliveira, Tomás Rodrigues
DETI, Universidade de Aveiro
Aveiro, Portugal
{ruipedrooliveira, tomasrodrigues}@ua.pt

Resumo –

Pretende-se através deste relatório expor sob forma escrita, o nosso desempenho e objetivos alcançados na aula prática n.5 da unidade curricular de Visão por Computador do Mestrado Integrado de Engenharia de Computadores e Telemática.

Neste relatório pretenderemos explicar as soluções por nós encontradas para a resolução dos diferentes problemas propostos.

***Palavras chave* – visão, computador, imagem digital, opencv, c++,**

I. REPOSITÓRIO: CÓDIGO FONTE

Todas as soluções dos problemas propostos estão disponível através do seguinte repositório (gitHub) criado para o efeito.

http://github.com/toomyy94/CV1617-68779-68129

A resolução dos problemas do presente guia encontram-se na pasta aula5. Para a resolução dos diferentes exercícios foi usado o `Code::Blocks IDE`.

II. PROBLEMAS PROPOSTOS

A. *Problema #1 - Chessboard calibration*

A.1 Enunciado

Compile and test the file `chessboard.cpp`. This code detects corners in a chessboard pattern using openCV functions and shows the results of the detection for a series of images. Use the available code to calibrate the camera used in the provided images (`left01.jpg` to `left13.jpg`). You need to use the function `calibrateCamera`.

A.2 Resolução e principais conclusões

Inicialmente, para a compilação deste exercício foi necessário a linkar a seguinte biblioteca: `libopencv_calib3d.so`
Para a calibração da câmara foi usada a seguinte função:

Listing 1: Função calibrateCamera

```
calibrateCamera(object_points, image_points,
    image.size(), intrinsic, distCoeffs, rvecs,
    tvecs);
```

Também para este exercício foram adicionados alguns prints de modo a que seja possível observar o valor das variáveis intrinsics, distortion, translations e rotations.
Por fim, este exercício permite ainda gerar um ficheiro XML com os valores de intrinsics e distortion.

Listing 2: Guardar intrinsics e distortion em XML

```
cv::FileStorage fs("../CamParams.xml",
    cv::FileStorage::WRITE);
fs << "cameraMatrix" << intrinsic <<
    "distCoeffs" << distCoeffs;
fs.release();
```

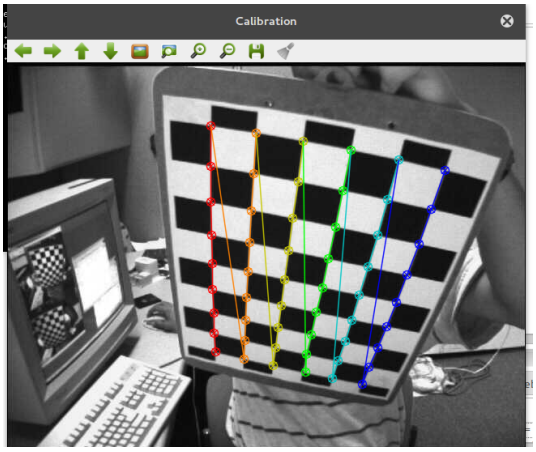

Figura 1: Resultado obtido após exercicio 1

B. *Problema #2 - Projection of 3D points in the image*

B.1 Enunciado

Use the function `cvProjectPoints2()` to project a line orthogonal (normal) to the chessboard (or a wireframe cube) in the provide images into each of the chessboard images using the rotation and translation vectors from the calibration.

B.2 Resolução e principais conclusões

Inicialmente, criámos um vetor do tipo `Point3f` que permitirá guardar os vértices do cubo. Foi usada a função `push_back()` para adicionar cada vértice ao vetor.

Listing 3: Criação do vetor para Proposta de resolução do exercício 5

```
std::vector<Point3f> new_object_points;
new_object_points.push_back(Point3f(0.0, 0.0,
    0.0)); // vertice0
// ... continue
```

Seguidamente, foi utilizada a função `projectPoints` com o seguinte atributos para a criação de uma projeção do cubo.

Listing 4: Criação do vetor para Proposta de resolução do exercício 5

```
projectPoints(new_object_points, rvecs.at(i),
    tvecs.at(i), intrinsic, distCoeffs,
    projected_points);
```

Finalmente, com a ajuda da função `line()` foram criadas as linhas do cubo com a cor amarela.

Listing 5: Desenhar linhas do cubo

```
line(image, projected_points[0],
    projected_points[1], Scalar(0, 255, 255), 2,
    8);
line(image, projected_points[0],
    projected_points[2], Scalar(0, 255, 255), 2,
    8);
// ... continue
```

As duas figuras seguinte ilustram o resultado do exercício 2.

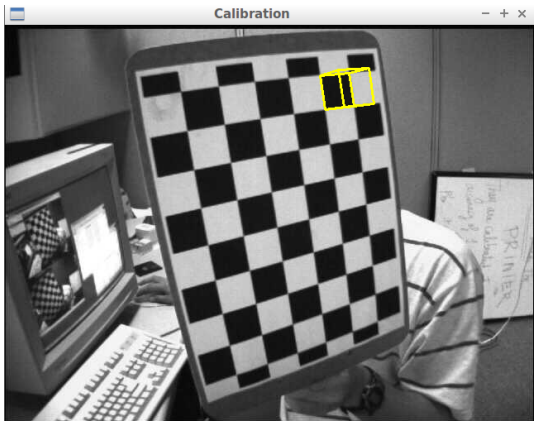

Figura 2: Resultado obtido após exercício 2

C. *Problema #3 - Using camera on computer*

C.1 Enunciado

Modify the code to use the camera from your computer to process the chessboard (comment the code for reading the provided images to allow switching between camera and provided images). Calibrate you camera with several chessboard images (use `cvWaitKey()` to move the chessboard position and a pre-defined number of images, for example 10). Be careful to check if the available chessboard is similar to the one in the provided images. If not, modify the code accordingly. If you want real metric distances, you need to update the code with the real distances of the used chessboard. Save the calibration parameters to a file.

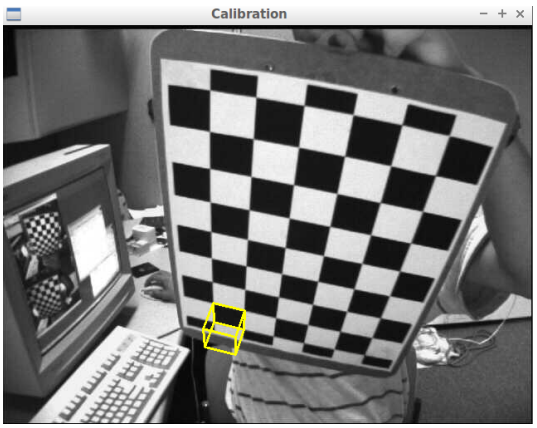

Figura 3: Resultado obtido após exercício 2

C.2 Resolução e principais conclusões

Este exercício permitirá receber imagem proveniente da câmara. Para tal, foi usada a classe `VideoCapture`.

Sempre que o utilizador pressiona a tecla *space*, um nova imagem é guardada. Quando são gravadas 13 imagens, o utilizador poderá ver a projeção resultantes, com o desenho do cubo tal como no exercício anterior.

As duas imagens seguinte, representam a receção da imagens proveniente da câmara.

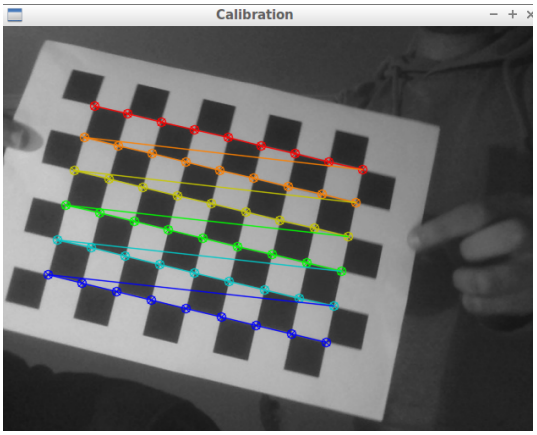

Figura 4: Resultado obtido após exercício 3

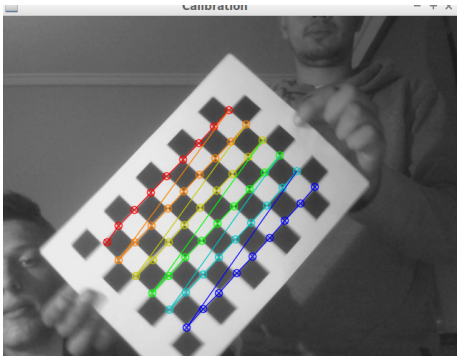

Figura 5: Resultado obtido após exercício 3

A imagem seguinte, representa a projetação do cubo.

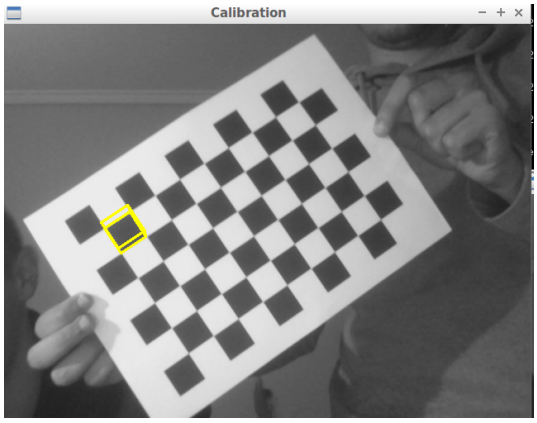

Figura 6: Resultado obtido após exercício 3

D. Problema #4 - External calibration

D.1 Enunciado

Calibrate a camera (using the given images or using your computer camera) and save the camera parameter file with another name. Modify the previous examples to read the intrinsic and distortion parameters from the file and perform external parameters calibration (using function `solvePnP` *) for a single image with the calibration pattern.*

D.2 Resolução e principais conclusões

Neste exercício calibramos a câmara com os parâmetros extrínsecos vindos do ficheiro `CamParams.xml`
Durante a captura de imagem pela câmara, esta foi convertida para cinzento e posteriormente foi aplicado a função `findChessboardCorners` de modo a obter os `corners` resultantantes.

Listing 6: Converter para cinza e calcular corners

```
cvtColor(image,image,CV_BGR2GRAY);
findChessboardCorners(image, board_sz,
    corners,0);
```

Listing 7: Leitura do XML

```
cv::Mat intrinsic_matrix;
cv::Mat distortion_coeffs;
cv::FileStorage fs("../CamParams.xml",
    cv::FileStorage::READ);
if (!fs.isOpened()){
   std::cerr << "Failed to open " << std::endl;
   return 1;
}
fs["cameraMatrix"] >> intrinsic_matrix;
fs["distCoeffs"] >> distortion_coeffs;
fs.release();
```

A sequência seguida para a resolução deste exercício foi a seguinte:

- Conversão da imagem obtida pela câmara para `CV_BGR2GRAY`
- Utilizar o método `findChessboardCorners()` para detetar os corners
- Utilização do método `solvePnP()`
- Utilização do método `projectPoints()`
- Desenhar linhas do cubo

O resultado obtida do exercício 4 encontra-se representado na figura seguinte.

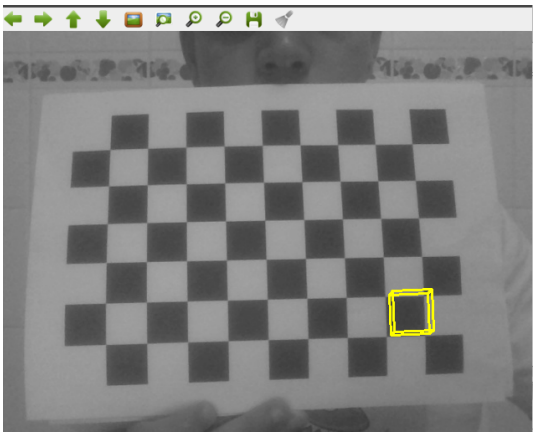

Figura 7: Resultado obtido após exercício 3

REFERÊNCIAS

[1] Neves, A. J. R.; Dias, P. Slides teóricos Visão por Computador - Aula 5 (2016)

[2] OpenCV. Opencv Documentation. Web. 15 Outubro 2016.