Visão por Computador 2016-17, Guia Prático N.º 6

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.º 6 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, stereo calibration, opencv, c++,**

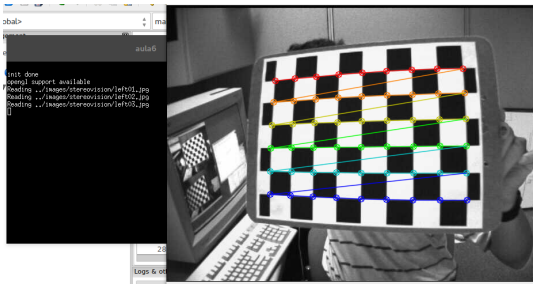

Figura 1: Resultado obtido após exercício 1

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 aula6. Para a resolução dos 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 (similar to the one used in the last lecture). Fill the necessary matrices with the correct value to calibrate the stereo pair. Analyze the code for filling the 3D coordinates of the pattern.

A.2 Resolução e principais conclusões

Efetuando umas ligeiras alterações ao código para o nosso IDE conseguimos compilá-lo e detetamos os cantos com a função `FindAndDisplayChessboard` da seguinte forma:

Listing 1: Função contagem dos cantos

```
// find and display corners
corner_count =
    FindAndDisplayChessboard(image,board_w,
      board_h,&corners);
```

B. Problema #2 - Stereo calibration

B.1 Enunciado

Calibrate the stereo pair using the function cvStereoCalibrate. After a successful calibration, save the matrices in an xml file using the file storage functions to avoid recalibration of the stereo rig each time.

B.2 Resolução e principais conclusões

Primeiramente calibramos a câmara para cada par de duas imagens dadas desta maneira:

Listing 2: Função streoCalibrate

```
Mat R, T, E, F;

double rms = stereoCalibrate(object_points,
    image_pointsL, image_pointsR,
cameraMatrix1, distCoeffs1,
cameraMatrix2, distCoeffs2,
imageL.size(), R, T, E, F,
TermCriteria(CV_TERMCRIT_ITER+CV_TERMCRIT_EPS,
    50, 1e-6),
CV_CALIB_FIX_FOCAL_LENGTH );
```

De seguida gravámos os parâmetros indicados no XML com o código fornecidoe escrevemos a matriz associada com os seus valores intrisecos, como se pode ver na imagem seguinte:

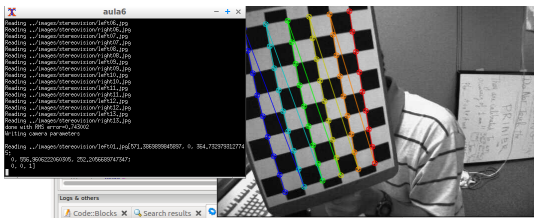

Figura 2: Resultado obtido após exercício 2

De seguida, encontra-se um excerto do ficheiro XML gerado.

Listing 3: Ficheiro XML gerado

```
<?xml version="1.0"?>
<opencv_storage>
<cameraMatrix_1 type_id="opencv-matrix">
  <rows>3</rows>
  <cols>3</cols>
  <dt>d</dt>
  <data>
    5.7138698998458972e+02 0.
        3.6473297931277449e+02 0.
    5.5696062220603051e+02 2.5220566897473466e+02
        0. 0. 1.</data></cameraMatrix_1>
<distCoeffs_1 type_id="opencv-matrix">
  <rows>1</rows>
  <cols>5</cols>
  <dt>d</dt>
  <data>
    -2.5451629689067001e-01 1.9279241569751074e-01
    -1.6798462796271199e-03 2.3027436119502927e-02
    -1.2838981716204551e-01</data></distCoeffs_1>
<cameraMatrix_2 type_id="opencv-matrix">
  <rows>3</rows>
  <cols>3</cols>
  <dt>d</dt>
  <data>
    5.5994944648413571e+02 0.
        3.5082756593670024e+02 0.
    5.6302127613737832e+02 2.3385777469609181e+02
        0. 0. 1.</data></cameraMatrix_2>
<distCoeffs_2 type_id="opencv-matrix">
  <rows>1</rows>
  <cols>5</cols>
  <dt>d</dt>
  <data>
    -4.8570573185961191e-01 1.7475908406455720e+00
    4.9687215534607008e-04 1.3814048691405257e-03
    -3.9846820537322838e+00</data></distCoeffs_2>
```

C. *Problema #3 - Lens distortion*

C.1 Enunciado

In a new file, read the distortion parameters of the cameras (function cvLoad), select a stereo pair of images from the pool of calibration images and present the undistorted images (image with the lens distortion removed) using the function cvUndistort to compute the new images.

C.2 Resolução e principais conclusões

Neste exercício é pedido para retirar a distorção produzida pelas lentes da câmara. Após gravar os parâmetros no XML como referido acima no exercício 2 usamos a função `cvUndistort` com o código abaixo mostrado, produzindo o seguinte resultado.

Listing 4: Função undistrort

```
Mat imageUndistortedL;
for (i=0;i<1;i++){
   // read image
   sprintf(filename,"../images/stereovision/
      left\%02d.jpg",i+1);
   printf("\nReading \%s",filename);
   imageL = cv::imread(filename,
       CV_LOAD_IMAGE_COLOR);

   undistort(imageL, imageUndistortedL,
       cameraMatrix1, distCoeffs1);

}
std::cout << cameraMatrix1 << std::endl;

imshow("output", imageL);
imshow("undistorted", imageUndistortedL);
```

Como conclusão consegue-se observar na imagem sem distorção das lentes que em especial as bordas do `chessboard` parecem muito mais retas, semelhantes à realidade. Vemos que na imagem do lado estas mesmas bordas têm uma saliência curva o que já quase não se verifica na imagem sem distorção (imagem à direita).

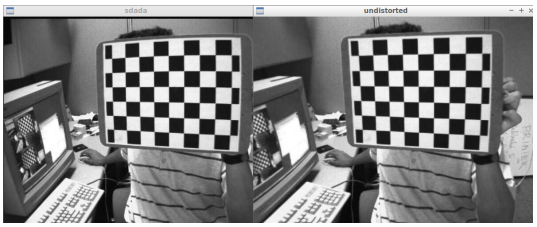

Figura 3: Resultado obtido após exercício 3

D. *Problema #4 - Epipolar lines*

D.1 Enunciado

Modify the previous example to show only the undistorted images. Add the possibility to select a pixel in each image using the following code to set a callback to be called for handling mouse events.

D.2 Resolução e principais conclusões

Temos um *callback* para determinar as coordenadas dos píxeis da imagem.

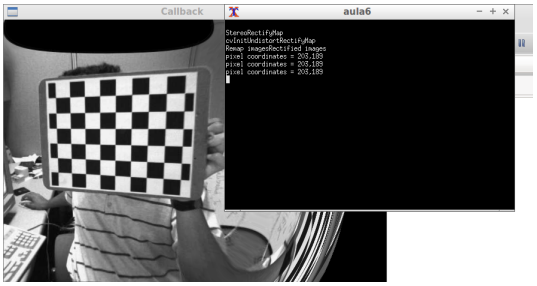

Figura 4: Resultado obtido após exercício 4

E. Problema #5 - Image rectification

E.1 Enunciado

Select a pair of stereo images and use the following OpenCV functions to generate the rectified images (corresponding epipolar lines in the same rows in both images). Visualize the resulting images, and draw lines in rows (for example at each 25 pixels) to evaluate visually if corresponding pixels are in corresponding lines.

E.2 Resolução e principais conclusões

Usámos dois *callbacks* neste exercício para determinar as coordenadas dos píxeis da imagem da esquerda e direita. Distinguindo-os como se pode ver na imagem abaixo.

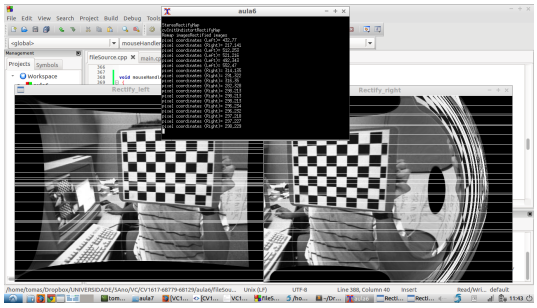

Figura 5: Resultado obtido após exercício 5

As linhas são desenhadas com recurso à função `cv:line` na imagem `rectified` correspondente.

Listing 5: Função mouseHAndler

```
void mouseHandlerL(int event, int x, int y, int
    flags, void* param){
   switch(event){
      case CV_EVENT_LBUTTONDOWN:
      cv::Mat* image = (cv::Mat*) param;
      std::cout << "pixel coordinates (Left)= "
          << x << "," << y << std::endl;
      std::vector<cv::Point2f> point;
      cv::line(*image,cvPoint(0,y),cvPoint(image
         ->size().width,y),cvScalar(255), 1, 8,
             0);
      cv::imshow("Rectify_right",*image);
      break;
   }
}
```

Todo o processamento anterior é feito desta maneira, tanto para a imagem da esquerda como para a da direita.

Listing 6: Função Rectify e remap

```
std::cout << "\nStereoRectifyMap";
cv::stereoRectify(intrinsics1, distortion1,
    intrinsics2, distortion2, imagel.size(),
    rotation, translation,R1, R2, P1, P2, Q,0);

std::cout << "\ncvInitUndistortRectifyMap";
cv::initUndistortRectifyMap(intrinsics1,
    distortion1, R1, P1, imagel.size(),
    CV_32FC1, map1x, map1y);
cv::initUndistortRectifyMap(intrinsics2,
    distortion2, R2, P2, imagel.size(),
    CV_32FC1, map2x,map2y);

std::cout << "\nRemap images";
cv::Mat gray_imagel;
cv::Mat remap_img1;
cv::cvtColor(imagel, gray_imagel , CV_RGB2GRAY);
cv::remap(gray_imagel, remap_img1, map1x,
    map1y,cv::INTER_LINEAR, cv::BORDER_CONSTANT,
    cv::Scalar());
```

REFERÊNCIAS

[1] Neves, A. J. R.; Dias, P. Slides teóricos Visão por Computador - Aula 6 (2016)

[2] OpenCV. Opencv Documentation. Web. 5 Novembro 2016.