Visão por Computador 2016-17, Guia Prático N.º 7

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.º7 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++,**

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 aula7. Para a resolução dos exercícios não foi usado nenhum IDE. Para a compilação do código fonte foi usada uma makefile.

II. PROBLEMAS PROPOSTOS

A. *Problema #1 - Disparity map*

A.1 Enunciado

Recover the code from the exercise on image rectification (exercise 5 in last lecture - in alternative you might use the available code in reconstruct.cpp). Use the class StereoBM and the function that implements a block matching techniques (template matching will be explored later within this Computer Vision course) to find correspondences over two rectified stereo images. Use the parameters specified as follow since we will not enter in details of these functions.

A.2 Resolução e principais conclusões

Para a resolução deste exercício foram seguidos os seguintes passos:

1. Ler parâmetros do ficheiro `stereoParams.xml` produzido na aula 6
2. Ler imagens stereovision cedidas - imagem da esquerda e direita
3. Aplicar métodos `stereoRectify` aos parâmetros intrinsics1, intrinsics2 , intrinsics1 e intrinsics2 com o objetivo de obter os parâmetros de rotação, translação...
4. Aplicar método `initUndistortRectifyMap` para obter variáveis `map1x`, `map1y`, `map2x` e `map2y`
5. Aplicar `cv::cvtColor(imagel, gray_imagel , CV_RGB2GRAY);`
6. Aplicar `cv::remap(gray_imagel, remap_imgl, map1x, map1y,cv::INTER_LINEAR);`
7. Aplicar os dois item anteriores à imagem da direita.
8. Aplicar o excerto de código disponível no enunciado desta aula.
9. Visualizar o resultado da disparidade.

O resultado obtidos pode ser observado na seguinte figura.

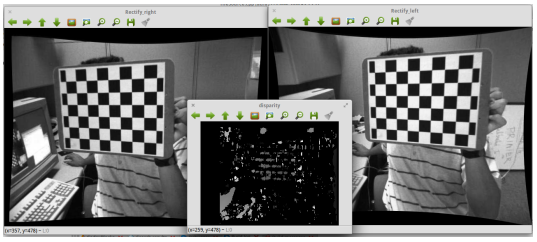

Figura 1: Resultado obtido após exercício 2

B. *Problema #2 - 3D Reconstruction*

B.1 Enunciado

Use the function cvReprojectImageTo3D to compute the 3D coordinates of the pixels in the disparity map. The parameters of cvReprojectImageTo3D are the disparity map (disp in previous exercise), and the matrix Q given by the function cvStereoRectify. Save the 3D coordinates in an xml file using the function FileStorage

B.2 Resolução e principais conclusões

Este exercício resultou de uma continuação do exercício anterior. No final desse, foi invocada a função `reprojectImageTo3D` com a variável resultante da função `imgDisparity16S.convertTo()`. Posteriormente foi gerado um ficheiro XML resultante da matrix `points3d`.

Listing 1: Aplicação da função reprojectImageTo3D e criação do ficheiro XML

```
Mat points3d;
reprojectImageTo3D(imgDisparity16S, points3d, Q);

cv::FileStorage fs2("../points3d.xml",
    cv::FileStorage::WRITE);
if (!fs.isOpened()) {
   std::cout << "Failed to open
       stereoParams.xml" << std::endl;
   return 1;
}
fs2 << "points" << points3d;
fs2.release();
```

Listing 2: points3d.xml

```
<?xml version="1.0"?>
<opencv_storage>
<points type_id="opencv-matrix">
  <rows>480</rows>
  <cols>640</cols>
  <dt>"3f"</dt>
  <data>
    6.71058960e+01 5.05488701e+01 -9.15517426e+01
        6.68978424e+01
    5.05488701e+01 -9.15517426e+01 6.66897888e+01
        5.05488701e+01
    -9.15517426e+01 6.64817276e+01 5.05488701e+01
        -9.15517426e+01
    6.62736740e+01 5.05488701e+01 -9.15517426e+01
        6.60656128e+01
    5.05488701e+01 -9.15517426e+01 6.58575592e+01
        5.05488701e+01
    -9.15517426e+01 6.56495056e+01 5.05488701e+01
        -9.15517426e+01
...
```

C. *Problema #3 - Visualization of point cloud in pcl*

C.1 Enunciado

Modify the source code viewcloud.cpp to read the 3D points of the file you have saved in the previous section and visualize the results of the 3D reconstruction. Assignment to the pointCloud (cloud in the example below) can be performed using the following code (or similar):

Listing 3: Excerto de código (enunciado)

```
int p=0;
for (int i=0;i< (*cloud).height; i++)
for (int j=0;j< (*cloud).width; j++)
{
(*cloud).points[p].x =
    pos3D.at<cv::Vec3f>(i,j)[0];
(*cloud).points[p].y =
    pos3D.at<cv::Vec3f>(i,j)[1];
(*cloud).points[p].z =
    pos3D.at<cv::Vec3f>(i,j)[2];
p++;
}
```

Visualize the 3 points and add any filtering necessary to avoid visualization of not well reconstructed 3d Points.

C.2 Resolução e principais conclusões

Começámos por ler o ficheiro `points3d.xml` resultante do exercício anterior. O resultado da leitura foi colocado na variavel `pos3D`. Posteriormente, as variáveis `width`, `height` e `is_dense` foram inicializadas da seguinte forma:

Listing 4: Inicialização de variáveis

```
(*cloud).width = pos3D.size().width;
(*cloud).height = pos3D.size().height;
(*cloud).is_dense = false;
(*cloud).points.resize((*cloud).width *
    (*cloud).height);
```

Seguidamente, foi utilizado o excerto de código que consta no enunciado deste exercício.

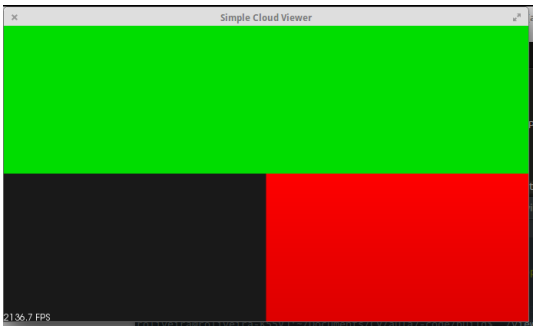

Figura 2: Resultado obtido após exercício 3

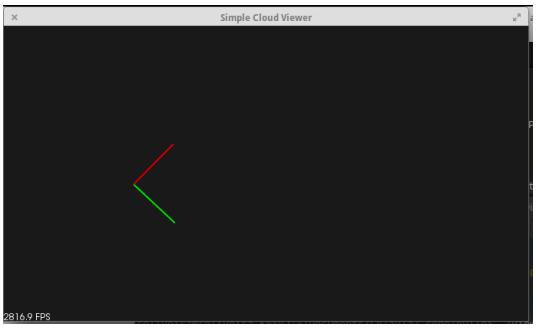

Figura 3: Resultado obtido após exercício 3

D. *Problema #4 - PCD (point cloud data) 3D format*

D.1 Enunciado

Modify the source code viewcloud.cpp to read and visualize the two provided kinect images filt_office1.pcd and filt_office2.pcd The Point Cloud Data file format (PCD) used is the 3D file format from PCL and can be written and read directly using the PCL functions loadPCDfile and savePCDFileASCII.

D.2 Resolução e principais conclusões

Para a resolução deste exercício foram criados dois variáveis do tipo `PointCloud<PointXYZRGB>::Ptr` : `cloud1` e `cloud2`. Posteriormente foram carregados os ficheiros `filt_office1.pcd` e `filt_office2.pcd` respetivamente para as variáveis `cloud1` e `cloud2`. Finalmente, foi usado o método `showCloud()` para visualização. O resultado obtido pode ser observado nas seguintes imagens.

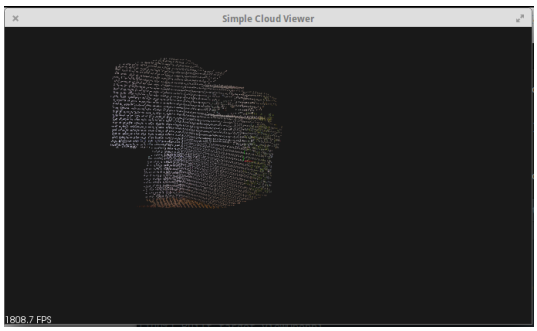

Figura 4: Resultado obtido após exercício 4

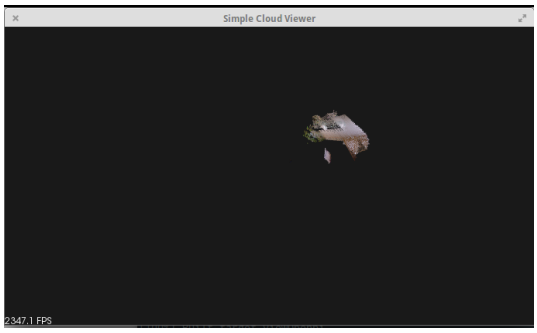

Figura 5: Resultado obtido após exercício 4

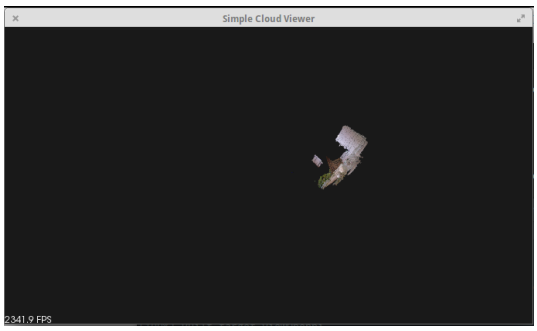

Figura 6: Resultado obtido após exercício 4

E. Problema #5 - ICP alignment

E.1 Enunciado

Use the pcl::IterativeClosestPoint function to align the given down sampled cloud of points. Note that the values of the termination criteria are very sensitive and should be adapted for each case. Normally an initial rough registration should also be provided to avoid bad registration However in this case, and given the proximity of the provided depth images this should not be necessary.

E.2 Resolução e principais conclusões

descricao
em falta

Listing 5: Função contagem dos cantos

```
code sample
```

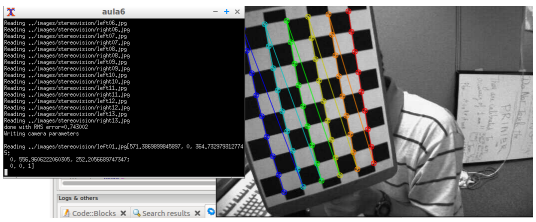

Figura 7: Resultado obtido após exercício 5

REFERÊNCIAS

[1] Neves, A. J. R.; Dias, P. Slides teóricos Visão por Computador - Aula 7 (2016)

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