Visão por Computador
07 - 3D Vision

Diogo Corte, Diogo Silva
DETI, Universidade de Aveiro
Aveiro, Portugal
{diogo.corte, dbtds}@ua.pt

***Resumo* – Este relatório foi realizado para a unidade curricular de Visão por Computadores com o objetivo de demonstrar a perceção da aula (VC Exercises 07). Sendo que o objetivo principal é imagens 3D, abordando mapa de disparidade, reconstrução 3D, visualização / obtenção imagens em formato PCD e alinhamento de imagens 3D.**

NOTA SOBRE AS IMAGENS

Apesar de as imagens terem aparência pequena, dispõem de boa resolução sendo que se for aplicado zoom sobre a imagem é possível ver com detalhe que se pretende.

I. DESCRIÇÃO DOS FICHEIROS

O conteúdo da pasta src contém os seguintes ficheiros:

1. reconstruct.cpp que é o responsável pela reconstrução 3D e exportação da disparidade para um ficheiro;
2. viewreconstruct.cpp que é o que permite visualizar o ficheiro exportado pelo ficheiro reconstruct.cpp numa point cloud;
3. viewcloudfile.cpp representa N ficheiros recebidos do formato PCD na mesma point cloud com a opção de ativar ou desativar o VoxelGrid;
4. viewcloudicp.cpp permite fazer a correção entre ficheiros do tipo PCD;
5. pcl_io.cpp permite fazer obtenção de imagens da Kinect com recurso ao OpenNI Grabber.

II. PROCESSO DE RECONSTRUÇÃO 3D A PARTIR DE IMAGENS STEREO

A. Imagem usada para a reconstrução

A imagem 1 foi a imagem escolhida para a reconstrução 3D.

Figura 1: Imagem rectificada usada para efeitos de reconstrução 3D

Sendo que é apenas uma imagem retificada, ou seja, as linhas epipolares de ambas as imagens coincidem, podendo retirar uma relação direta entre a imagem da esquerda e a imagem da direita.

B. Mapa de disparidade

Com o par stereo (imagens esquerda e direita) rectificado e em escala de cinzentos, é possível encontrar correspodências entre o par. Para o calculo do mapa de disparidades o OpenCV fornece a class cv::StereoBM, onde é possível definir o range de disparidade e o tamanho da janela de matching.

Figura 2: Mapa de disparidade da imagem 1

Na figura acima está representado o mapa de disparidades obtido para a Figura 1, em que o significado dos valores obtidos são que areas a escuro (valores de cor a tender para 0) são areas de muita diferença entre o par, e areas a branco (valores de cor a tender para 255) representam pouca diferença. Temos assim que o tabuleiro de xadrez, o computador/teclado e a camisa da pessoa em cena são os pontos que se encontravam mais próximos entre as duas imagens.

C. Função ReprojectImageTo3D

O método ReprojectImageTo3D transforma a matriz de disparidades (apenas um canal) obtida anteriormente, numa imagem de 3 canais com a representação da superfície 3D. Sendo assim possível relacionar qualquer ponto da imagem 2D rectificada da esquerda (é a referência da reconstrução) com um ponto 3D.
No projeto elaborado serializou-se para ficheiro (Image3D_Reconstructed.xml) alguns dados úteis, sendo armazenada a imagem reconstruida e a imagem resultante da concatenação pela ordem natural (esquerda para a direita), das imagens retificadas.

D. Visualização da Reconstrução 3D na PointCloud

Para a visualização dos resultados obtidos pela função ReprojectImageTo3D o projecto contem o binário viewReconstruct, que faz o display dos dados armazenados em Image3D_Reconstructed.xml. A representação do ficheiro exportado para a point cloud é directa, tendo em conta que o ficheiro exportado já contém uma matrix de pontos 3D.

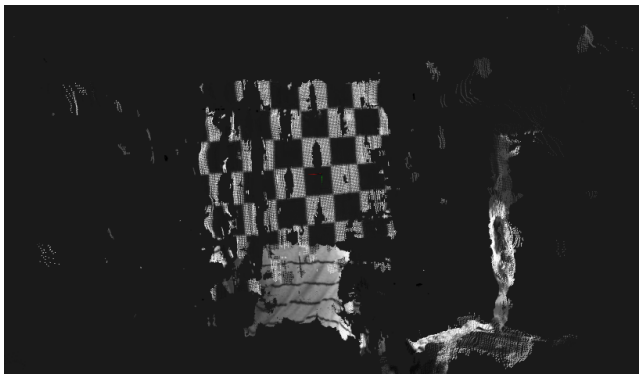

Figura 3: Visualização do resultado da disparidade na PointCloud

A imagem reprojectada para 3D obtida reflete os resultados obtidos no ponto B, onde as superfícies representadas foram o tabuleiro de xadrez, o computador/teclado e a camisa da pessoa em cena, tal como já acontecia no mapa de disparidade.

D.1 Representação RGB na Reconstrução 3D

Para além de pegarmos nos pontos e imprimi-los na point cloud, ainda se representou as cores dos mesmos, sendo que ajuda bastante na perceção do cenário. Para isso foi necessário ao exportar a imagem no reconstruct, enviar também a imagem original, para poder fazer a associação do ponto 3D, ao ponto real da imagem 2D retificada.

A imagem que é usada como referência para a reconstrução 3D é a da esquerda da rectificação, sendo essa a que foi usada para seleccionar as cores dos pixeis.

```
pcl::PointCloud<pcl::PointXYZRGB>::Ptr cloud(new
    pcl::PointCloud<pcl::PointXYZRGB>);
..
cloud->points.at(p).r =
    image.at<cv::Vec3b>(i,j).val[0];
cloud->points.at(p).g =
    image.at<cv::Vec3b>(i,j).val[1];
cloud->points.at(p).b =
    image.at<cv::Vec3b>(i,j).val[2];
```

No trecho que código é possível ver as alterações que foram efetuadas para colocar esta situação a funcionar, sendo que a point cloud passou a ser do tipo PointXYZRGB em vez de PointXYZ, recebendo mais 3 argumento, R, G e B.

III. OBTENÇÃO DE PCD ATRAVÊS DA KINECT

Na imagem 4 que foi fornecida é possível visualizar que contém armários, pequenas estantes para livros, trepadeiras, ou seja, é possível perceber o contexto da cena com uma imagem retirada através da Kinect. Esta imagem foi representada recorrendo a PointCloud que permite a representação de estruturas 3D.

Figura 4: Representação de uma das imagens (office1) no PointCloud

No entanto apenas está representado umas das suas imagens sendo que ainda foi aumentado o point size 2 vezes de forma a tirar a abstração de pontos e parecer que se está sobre a existência de um conjunto de planos representados em 3D e não de um conjunto de pontos. Sendo que estas imagens podem ser obtidas através do OpenNI Grabber.

A. Representação de duas imagens da Kinect

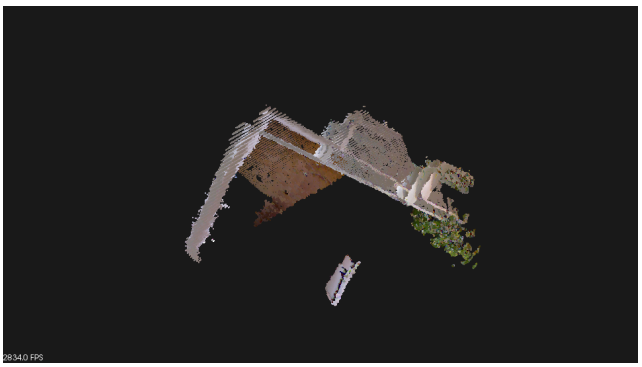

Figura 5: Representação de duas imagens não alinhadas num PointCloud

Se é possível representar numa PointCloud uma imagem, também é possível reproduzir outra imagem que foi retirada pela Kinect na mesma PointCloud, isto faz sentido quando se pretende relacionar as imagens de forma a obter uma melhor perceção do cenário em questão, neste caso, o escritório onde foram retiradas as imagens da Kinect.

No entanto ao representar as duas imagens na mesma PointCloud surgiu um problema, a imagem 5 mostra uma perspetiva planta a partir de baixo do escritório, sendo possível ver os limites das paredes e o problema em questão.
Na imagem 5 pode-se verificar que apesar de ambas as imagens estarem representadas na mesma PointCloud com sucesso, o canto do escritório que devia coincidir na PointCloud devido a serem o mesmo escritório, não coincide. Isto pode-se ver no canto superior da imagem onde uma das paredes do escritório que foi representada está mais afastada que a outra em vez de coincidirem. Sendo necessário minimizar o erro entre a distância dos pontos correspondentes de cada imagem (III-C).

B. Utilização da Voxel Grid

Antes proceder ao alinhamento das imagens, convém que a representação de cada imagem esteja uniforme e que pequenas correções que sejam precisas nas imagens são efetuadas antes de aplicar o ICP. Por exemplo, a Kinect ao reproduzir os ficheiro PCD coloca vários pontos com NaN (Not a number) devido a não retirar informação suficiente sobre os mesmos, com o Voxel Grid consegue-se fazer uma filtragem a esses valores.

Na imagem 6 pode-se verificar o resultado da aplicação do VoxelGrid numa das imagens (office1).

Figura 6: Aplicando do filtro Voxel Grid na imagem office1.pcd

O Voxel Grid é um algoritmo que faz downspampling a amostra de pontos, sendo que ele considera uma grelha de tamanho fornecido (X,Y,Z) e que para cada grelha retira unicamente um ponto. Sendo esse ponto uma média de todos os pontos pertencentes a grelha ou apenas o centroide da grelha [1], sendo assim possível reduzir o número de amostras das imagens originais e filtrar os valores NaN para ser possível aplicar o alinhamento ICP.

C. Alinhamento ICP

O alinhamento ICP, Iterative Closest Point, é um algoritmo iterativo tal como o nome diz que com base em transformações de rotação e translação permite minimizar a distância entre duas imagens, sendo que recebe como parâmetros iniciais a estimativa da transformada e o critério de paragem do algoritmo [2].

Na imagem 7 é possível ver o resultado da aplicação do algoritmo ICP.

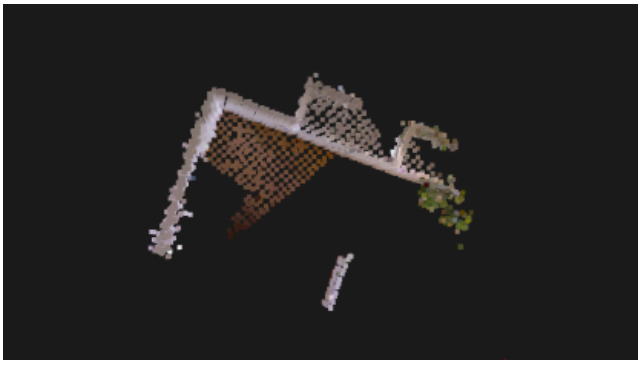

Figura 7: Representação de ambas as imagens após aplicar o ICP

Na imagem é possível verificar que os cantos já se encontram alinhados (ao contrário do que apresentado anteriormente), sendo que as duas imagem fornecidas foram representadas uma em função de outra na mesma PointCloud. Era possível continuar a fazer a representação de mais imagens na mesma PointCloud, sendo que bastava colocar o resultado do ICP como source para o próximo ICP juntamento com a imagem pretendida no alinhamento.

D. Aquisição de imagens de profundidade utilizado a Kinect

D.1 Imagens obtidas

Note-se a imagem 8, uma das imagens que foi obtidas durante o uso da Kinect:

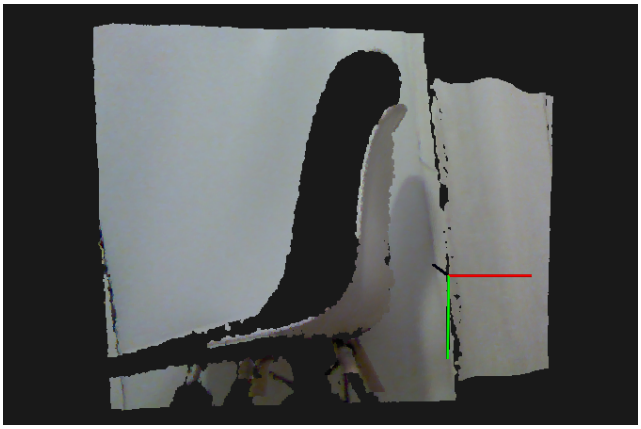

Figura 8: Obtenção de uma cadeira utilizando a Kinect

Na cadeira é possível observar a curvatura da mesma na parte de trás, sendo que a cadeira se apresenta de lado. Também é possível visualizar as pernas da mesma. Para além da cadeira em si, também se pode reparar a cinzento claro uma sombra que a cadeira faz na parede.
No entanto, note-se que existe uma zona preta projectada na parede que é a area que a cadeira está a cobrir da parede, sendo que a Kinect não consegue identificar o que está por trás da cadeira.

Nas secções seguintes é possível verificar os problemas e como foram resolvidos durante a instalação do OpenNI e do PCL.

D.2 Instalação na distribuição Ubuntu 16.04

Para a distribuição Ubuntu 16.04 usou-se o seguinte tutorial:

`https://larrylisky.com/2014/03/03/installing-pcl-on-ubuntu/`

Sendo que foi possível instalar e fazer aquisição das imagens de profundidade da Kinect.

D.3 Instalação na distribuição ArchLinux

A obtenção das imagens da Kinect exige a instalação do OpenNI e do PCL. Neste caso houve certos problemas encontrados durante a instalação do mesmo:

1. Erro na compilação das bibliotecas devido a compilação estar a ser efetuada com a versão 6 do GCC e do G++, as bibliotecas OpenNI e PCL apenas compilam com versões 5 ou inferior do GCC/G++.
2. Bibliotecas de OpenNI não estavam a ser encontradas após instalar ambos, isto deve-se a ter instalado primeiro o PCL e depois o OpenNI. Como o PCL só coloca as bibliotecas de manipulação do OpenNI no sistema se encontrar o OpenNI no sistema, daí o problema. Solução para este problema bastou instalar o OpenNI antes de instalar o PCL.
3. Ao executar os executáveis de teste do OpenNI dá erro em runtime com ”No matching devices”, sendo que este erro apenas deveria aparecer quando existe dispositivos detetados não compatíveis, que não é o caso da Kinect, inclusive no ficheiro /etc/udev/rules.d/55-primesense-usb.rules encontra-se devidamente identificado, sendo que as correspondências do ficheiro correspondem a resposta do lsusb que identifica o dispositivo Kinect sem problemas. Sendo que este problema não se conseguiu resolver.

Não foi possível fazer a aquisisão de imagens através da Kinect devido a último problema mencionado, no entanto, foram utilizadas as imagens fornecidas (office1.pcd e office2.pcd) para as soluções obtidas.

REFERÊNCIAS

[1] *Point Matcher Voxel Grid explanation*. Acedido a 14 de Novembro.

[2] *PCL Tutorial - Vision and Perceptual Machines Lab, Jeff Delmerico*. Acedido a 15 de Novembro.