Relatório Visão Por Computador
Kinect Based 3D Reconstruction System

Luís Silva, nmec 88888
luisfgbs@ua.pt
Mestrado em Robótica e Sistemas Inteligentes
Universidade de Aveiro

I. Introdução

Este relatório foi redigido no âmbito da UC Visão Por Computador, lecionada pelo professor Paulo Dias e tem o objetivo de documentar o trabalho realizado durante o decorrer do projeto. Este projeto pretende fazer uso das capacidades da Kinect e da biblioteca Open3d para proceder à reconstrução 3d de espaços.

Janeiro 2022

II. Kinect

A. Composição e imagens produzidas

A Kinect v1.0 é composta por uma câmara RGB e um sensor infra-vermelho (emissor e recetor). Esta configuração permite que para uma mesma cena o utilizador possa recolher informação relativa à cor e à profundidade dos elementos presentes na imagem.
As imagens produzidas têm dimensões de 640×480.
No ficheiro Setup encontram-se um guia elaborado para ajudar o utilizador a executar os passos necessários para estabelecer a conexão à Kinect

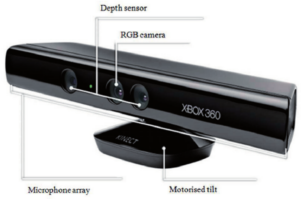

Fig. 1. Kinect v1.0 e seus componentes

Em cada instante podemos capturar as imagens RGB e de depth produzidas pela kinect recorrendo ao código apresentado nas figuras 2 e 3, respetivamente. Na prática, relativamente à imagem depth, a kinect produz um mapa de disparidades que pode ser normalizado para visualização representando as profundidades como intensidades de píxel (grayscale)

```
#function to get RGB image from kinect
def get_video():
    array,_ = freenect.sync_get_video()
    return array
```

Fig. 2. Excerto de código para a captura de imagens RGB produzidas pela kinect

```
#function to get depth image from kinect
def get_depth():
    array,_ = freenect.sync_get_depth()
    array = array.astype(np.uint8)
    return array
```

Fig. 3. Excerto de código para a captura de imagens depth produzidas pela kinect

B. Calibração

A calibração da câmara foi feita da forma já explorada nas aulas e no ”Relatório Visão Por Computador - Aula Prática 5” previamente submetido para avaliação

III. Open3d

Esta biblioteca foi-nos apresentada nas aulas práticas da UC e tendo funcionalidades completas e intuitivas de visualização, manipulação e interação com nuvens de pontos provou-se fulcral ao desenvolvimento do projeto

IV. Scripts

Os scripts referentes às funcionalidades exploradas e que devem ser usados pelo utilizador são os indicados em baixo, os restantes foram usados como suporte/exploração de conceitos e código em auxílio da criação dos principais

1) depth_to_cloud.py: Utilizado para capturas singulares de imagens e criação das respetivas nuvens de pontos

2) depth_to_cloud_live.py: Utilizado para a captura contínua de imagens e criação das respetivas nuvens de pontos

3) icp.py: Faz o registo inicial utilizando a matriz homogénea

4) mmc.py: A sigla mmc representa multiple manual correspondences e é utilizado portanto para fazer o registo inicial manualmente entre 2 ou mais nuvens de pontos

5) livefeed.py: Almeja conseguir fazer o processo de sobreposição das nuvens sem necessitar da interação do utilizador

V. Nuvem de pontos

Uma nuvem de pontos permite a atribuição dos parâmetros x, y e z (coordenadas cartesianas) a cada um dos pontos que a formam e permite também, assim, a representação a 3 dimensões de um espaço ou objeto.

A. Criação

Para a criação da nuvem de pontos sem informação de textura podemos apenas recorrer ao método PointCloud.create_from_depth_image() que recebe a depth image (convertida para um formato reconhecido pelo open3d) e os parâmetros intrínsecos da câmara para fazer a projeção dos pontos no espaço

```
rgb_img=o3d.geometry.Image(frame)
depth_img=np.float32(depth)
depth_map=o3d.geometry.Image(depth_img)
pcd_depth=o3d.geometry.PointCloud.create_from_depth_image(depth_map, intrinsics)
```

Fig. 4. Excerto de código para a criação de nuvem de pontos sem textura

B. Textura

Para adicionar textura, devemos primeiro criar uma imagem rgbd que junta a informação de cor com a informação de profundidade dos pontos da cena. Pode-se fazer utilizando o método RGBDImage.create_from_color_and_depth() que recebe a imagem RGB e depth capturadas. A documentação do open3d por vezes não é muito explícita mas pela exploração que fiz parece-me que esta combinação de informações é feita essencialmente por justaposição e atribuição do valor de cor ao vizinho mais próximo embora fosse de esperar que se fizesse uso dos parâmetros intrínsecos da câmara. A criação da nuvem de pontos é similar recebe apenas a imgem rgbd em vez de receber só a informação de profundidade

```
rgbd=o3d.geometry.RGBDImage.create_from_color_and_depth(rgb_img, depth_map, convert_rgb_to_intensity=False)
pcd_rgbd=o3d.geometry.PointCloud.create_from_rgbd_image(rgbd, intrinsics)
```

Fig. 5. Excerto de código para a criação de nuvem de pontos sem textura

VI. Iterative Closest Point (ICP)

Iterative Closest Point é o processo através do qual se faz a aproximação iterativa de duas nuvens de pontos. Em cada iteração procura-se a transformação que minimize as distâncias entre os pontos das nuvens.
É possível avaliar o resultado final, esta avaliação é feita sob 3 aspetos

1) Fitness: Mede a sobreposição das nuvens, quanto maior melhor

2) Inlier_rmse: Indica a probabilidade de, para cada correspondência encontrada, a correspondência não ser a correta, resumidamente é o erro. Quanto menor melhor

3) Correspondence set size: Número de correspondências encontradas

A. Registo

Para que seja possível aplicar point to point ICP deve-se primeiro efetuar o registo das nuvens, ou seja, aproximá-las por forma a que haja pontos de sobreposição a partir dos quais a função de avaliação do algoritmo possa calcular as novas tranformações. Seguem-se dois métodos distintos de efetuar este registo

1) Aproximação através de matriz homogénea: Consiste em, descrever os movimentos de translação e rotação da câmara entre duas capturas para que essa mesma transformação possa ser aplicada às nuvens.
Embora os resultados sejam satisfatórios, rapidamente se torna difícil a construção desta matriz, principalmente se a câmara rodar sobre mais de um eixo entre as capturas.

2) Aproximação manual: Consiste na marcação de pelo menos 3 pontos correspondentes nas duas nuvens para que através da sobreposição dos mesmos se estime a transformação apropriada para a aproximação das nuvens. Apesar de produzir resultados muito bons, rapidamente se torna numa tarefa extensa uma vez que para N nuvens o utilizar terá que fazer 2N marcações de pontos

VII. Livefeed

Apesar do sucesso dos métodos anteriores, o facto de necessitarem de input prévio ou em tempo de execução do utilizador barra a implementação da feature de livefeed. Para ultrapassar isto passa-se a implementar um novo processo de aproximação das nuvens que não requer a participação do utilizador, Global Registration

A. Global registration

a) Fast Point Feature Histograms - FPFH: Caracteriza cada ponto através da definição de um histograma que mapeia 33 parâmetros que são independentes da perspetiva como por exemplo as normais

1) Point-to-plane alignment: Utilizado como método de refinamento dos resultados obtidos com a aplicação de global registration. Faz uso das normais para uma melhor aproximação das nuvens de pontos

2) Fast Global registration: Esta processo aplica mecanismos próprios para aumentar a eficiência do código reconhecendo mais depressa quando as correspondências são inválidas

Em baixo apresenta-se organizado numa tabela a colheita de dados respetiva a avaliação do resultado da aplicação dos diferentes algoritmos de registo

Fig. 6. Nuvem de pontos resultante de global registration - vista frontal

Fig. 7. Nuvem de pontos resultante de global registration - vista de cima

Fig. 8. Nuvem de pontos resultante de global registration seguido de point to plane alignment- vista frontal

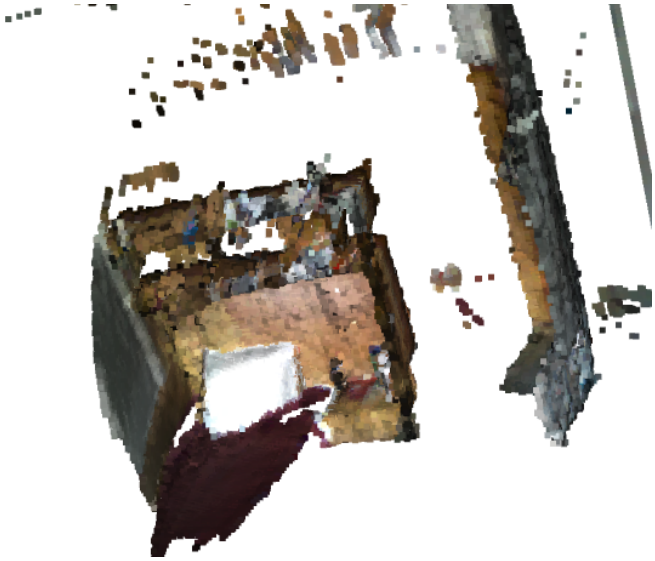

Fig. 9. Nuvem de pontos resultante de global registration seguido de point to plane alignment - vista de cima

VIII. Conclusão

As respostas encontradas aos desafios propostos integrar a kinect e o open3d, criar a nuvem de pontos e adicionar informação de textura, explorar diferentes algoritmos de registo e implementar a funcionalidade de livefeed foram bastante satisfatórias. Penso que o trabalho desenvolvido é suficiente como prova de conceito de que o programa seria já capaz de recriar ambientes sendo que a sua limitação é a memória pois rapidamente, se o espaço for muito grande, o carregamento das nuvens de pontos e a sua posterior atingem dimensões consideráveis

Voxel size	Fitness	Inlier rmse	Corresp	Stage
0.003	2.6×10^{-1}	1.09×10^{-3}	1631	Fast
0.003	3.3×10^{-1}	8.78×10^{-4}	102605	Post-alignment*
0.003	1.9×10^{-1}	2.72×10^{-3}	263	Pre-aligment