Trabalho Laboratorial 2 – Monitorização contra o COVID19

1. Introdução

Neste trabalho laboratorial consiste em várias tarefas, com o objetivo final de monitorizar o número de pessoas num laboratório. Para tal, irá utilizar um sensor Intel RealSense RGBD, nomeadamente o modelo D435, colocado por cima da entrada do laboratório, virada para baixo, como mostra a Figura 1. O objetivo é contar o número de pessoas dentro do laboratório, com o sensor a ser utilizado para contar o número de pessoas que entra e sai do laboratório. Dado que o sensor está a apontar para baixo, será mais difícil ver a cara das pessoas, logo o risco associado à proteção de dados pela identificação de indivíduos será muito menor – não se pretende distinguir as pessoas, mas somente determinar o número pessoas que está dentro da sala, isto é, o número de pessoas que entraram menos o número de pessoas que saíram.

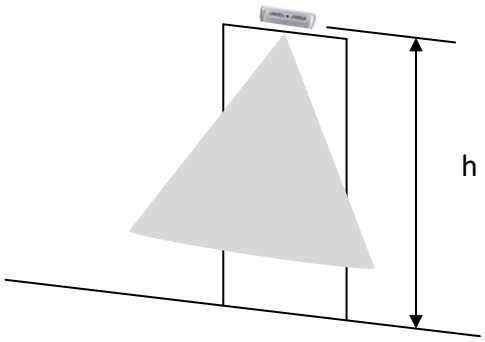

Figura 1 – Esboço da colocação do sensor por cima da entrada.

Este trabalho está dividido em duas tarefas obrigatórias e uma opcional:

> **Tarefa 1**: implementar o cálculo do mapa de profundidade usando visão estéreo.
> **Tarefa 2:** calcular o número de pessoas dentro do laboratório, através da determinação das pessoas que entram e saem do laboratório.
> **Tarefa 3** (opcional)**:** implementar a calibração do sistema estéreo.

Este documento está estruturado da seguinte forma: o restante desta secção descreve a Intel RealSense D435; a Secção 2 descreve o material fornecido para este trabalho, nomeadamente os ficheiros fornecidos e o seu conteúdo; a Secção 3 descreve a primeira tarefa deste trabalho; a Secção 4 descreve a segunda tarefa deste trabalho; a Secção 5 descreve a tarefa opcional que pode ser implementada para créditos extra; finalmente, a Secção 6 fornece algumas notas finais que poderão ser úteis ao desenvolvimento.

1.1 A Intel RealSense D435

Com o lançamento da família de dispositivos RealSense da Intel, ficaram disponíveis novos sensores RGBD de baixo custo e precisão considerável, permitindo uma utilização mais alargada destes dispositivos, principalmente em aplicações de robótica, produção,

monitorização e inspeção. A Figura 2 mostra uma vista exterior da Intel RealSense D435 a qual, medindo apenas 90 mm x 25 mm x 25 mm, incluí um sistema de visão estéreo ativo, composto por um projetor infravermelhos, duas câmaras infravermelhos e uma câmara RGB, como ilustrado na Figura 3. O sensor é alimentado através da ligação USB e possui um módulo ASIC de visão capaz de calcular mapas de profundidade com 1280x720 pixéis a 90 fps a distâncias entre 0.1m e 10 m, consumindo apenas 2 W.

Figura 2 – Visa exterior da Intel RealSense D435.

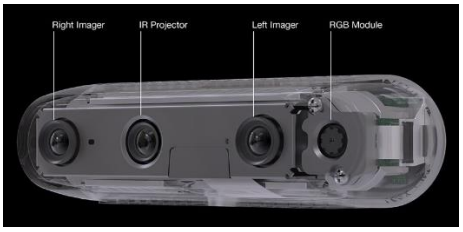

Figura 3 – Vista interior da Intel RealSenseD435.

Quando ligado a um sistema Windows ou Linux, o sensor é detetado como um dispositivo UVC (*USB Video device Class*), ficando imediatamente disponível para a utilização típica das câmaras. No entanto, a Intel disponibiliza o Intel RealSense SDK, um SDK multiplataforma que incluí *bindings/wrappers* para múltiplas linguagens, incluíndo Python. Ao utilizar o SDK tem acesso a cada um dos fluxos (*stream*) de vídeo individuais, quer da câmara RGB, que das duas câmaras infravermelhos, além de outras funcionalidades, como, por exemplo, o controlo do projeto infravermelhos. Mais abaixo serão dados mais detalhes sobre a utilização do SDK.

2. Configuração e material fornecido

Nesta secção descreve-se os requisitos necessários para instalar o SDK para trabalhar com os dados fornecidos, bem como os dados fornecidos.

Neste trabalho os dados são fornecidos principalmente na forma dum ficheiro *bag*. Embora pudéssemos ter fornecido os vídeos em ficheiros separados, ao fornecer os dados num ficheiro do tipo *bag*, será como se estivesse a obter os vídeos diretamente do sensor, incluindo informação temporal associada a cada aquisição. Desta forma não ser terá que preocupar co questões de sincronização da informação obtida das várias câmaras.

A Intel disponibiliza atualmente o Intel RealSense SDK 2.0, o qual suporta várias linguagens de programação, tais como C++, Javascript e Python, entre outras. Dado que estaos a usar o Python 3.7.x, o SDK pode facilmente ser instalado através do terminal do Windows usando o seguinte comando:

```
>> pip install pyrealsense2
```

É aconselhado usar o Python 3.7.x, como sugerido no início do semestre, de forma a poder utilizar o comando acima, caso contrário terá que instalar o SDK a partir do código-fonte. Relativamente à documentação do Intel RealSense SDK, este está disponível aqui, mas apenas deve consultar essa documentação se estiver interessado em detalhes adicionais, dado que abaixo irá encontrar toda a informação que necessita para ter acesso aos dados para este trabalho.

Com o SDK instalado, pode agora interagir com os dados fornecidos. Caso pretenda ter uma primeira visualização dos dados disponibilizados, pode abrir o Intel RealSense Viewer (incluído

na instalação). Depois, no Intel RealSense Viewer, clique em na opção *Add Source* no canto superior esquerdo, escolha a opção *Load Recorded Sequence*, escolha o ficheiro *bag*, e depois clique no botão para iniciar (*play*).

A lista abaixo descreve os ficheiros disponibilizados, quer os do tipo *bag*, quer os restantes (posteriormente neste documento será dada informação adicional, nomeadamente no que diz respeito à sua utilização):

CV_D435_20201104_160738_RGB_calibration.bag[1]**:** ficheiro *bag* que contém apenas dados da câmara RGB, enquanto é mostrado um padrão de xadrez em diferentes poses;

CV_D435_20201104_161043_Full_calibration.bag[1]**:** ficheiro *bag* com dados de todos as câmaras, enquanto é mostrado um padrão de xadrez, com o projetor de infravermelhos desligado.

CV_D435_20201104_162148.bag[1]**:** ficheiro *bag* completo com dados de todas as câmaras e com o projetor infravermelhos ativado

CV_D435_IR_(left|right)_01(.png|.raw|_metadata.csv): capturas das câmaras esquerda e direita infravermelhos, com o projetor ativado, em formato PNG e RAW. O ficheiro CSV contém informação sobre a captura, nomeadamente a resolução da câmara, os parâmetros intrínsecos e modelo de distorção utilizado (neste caso, o modelo de distorção Brown-Conrady).

CV_D435_IR_(left|right)_calib_Infrared(.png|.raw|_metadata.csv): semelhante à anterior, mas obtida com o projetor desativado e com o padrão xadrez colocado no chão.

CV_D435_RGB_calib_Color.png: semelhante à anterior, mas relativamente à câmara RGB.

Se está a utilizar o Microsoft Windows, o SDK da Intel RealSense é tipicamente instalado em `C:\Program Files (x86)\Intel RealSense SDK 2.0`. Aí encontra a pasta `samples` com muitos exemplos, mas que estão principalmente em C++. No github encontra um conjunto de exemplos adicionais em Python onde, entre outros, se encontra o exemplo read_bag_example.py, o qual mostra como pode obter os dados a partir dum ficheiro *bag* com dados da Intel RealSense. Descarregue o exemplo e, na linha 43, na chamada da função `enable_stream`, altere a resolução para 848x480 (2º e 3º argumentos, respetivamente), bem como a taxa de aquisição para 15 (5º argumento), de forma a coincidir com a informação que está guardada no ficheiro *bag* (o ficheiro *read_bag_example.py* fornecido já inclui estas alterações).

O exemplo que acabou de experimentar, mostra como pode obter os dados no formato de imagem do OpenCV (Numpy), que poderá utilizar para este trabalho. O exemplo está bem explicado através dos comentários incluídos, mas iremos descrever aqui as principais funções, de forma a melhor perceber como funciona a interação dos dados das câmaras da Intel RealSense.

```
    # Create pipeline
    pipeline = rs.pipeline()
```

[1] Os ficheiros *bag* são fornecidos como ligações devido ao seu tamanho (~400Mb, ~800Mb e ~3Gb, respetivamente). Todos os outros ficheiros estão incluídos no ficheiro comprimido do trabalho.

```
    # Create a config object
    config = rs.config()
    # Tell config that we will use a recorded device from file to be used
 by the pipeline through playback.
    rs.config.enable_device_from_file(config, args.input)
    # Configure the pipeline to stream the depth stream
    config.enable_stream(rs.stream.depth, 848, 480, rs.format.z16, 15)

    # Start streaming from file
    pipeline.start(config)
```

Código 1 – Inicialização da configuração do fluxo e aquisição de dados.

No Código 1 encontra a configuração típica e respetiva inicialização do fluxo de dados. A única diferença comparativamente à utilização da câmara real está na utilização do ficheiro *config* com a função *enable_device_from_file*, a qual não seria utilizada se estivéssemos a comunicar diretamente com a câmara. No sensor Intel RealSense cada câmara irá disponibilizar o seu fluxo de dados, mas existe um fluxo adicional de profundidade que pode ser disponibilizado pelo hardware bordo do sensor, calculado a partir dos dados do sistema estéreo infravermelhos. A função *enable_stream* pode se utilizada para configurar cada fluxo de dados e, neste caso em particular, está a ser utilizado para configurar o fluxo de dados do mapa de profundidade para usar uma imagem de 848 pixéis de largura, 480 pixéis de altura, no formato `rs.format.z16` (inteiro sem sinal de 16-bit por pixel), e uma taxa de aquisição de 15 fps. Note que apenas quando corre a função *pipeline.start(config)* é que a câmara começa de facto a fornecer os dados com a configuração desejada.

```
# Create colorizer object
    colorizer = rs.colorizer()

    # Streaming loop
    while True:
        # Get frameset of depth
        frames = pipeline.wait_for_frames()

        # Get depth frame
        depth_frame = frames.get_depth_frame()

        # Colorize depth frame to jet colormap
        depth_color_frame = colorizer.colorize(depth_frame)

        # Convert depth_frame to numpy array to render image in opencv
        depth_color_image = np.asanyarray(depth_color_frame.get_data())

        # Render image in opencv window
        cv2.imshow("Depth Stream", depth_color_image)
        key = cv2.waitKey(1)
        # if pressed escape exit program
        if key == 27:
```

```
            cv2.destroyAllWindows()
            break
```

Código 2 – Aquisição e exibição do mapa de profundidade.

No Código 2 encontra o código relativamente à obtenção do mapa de profundidade disponibilizado pelo sensor e sua exibição ao utilizador. Primeiro obtém-se um objeto colorizer, que será posteriormente utilizado para criar uma imagem a cores a partir do mapa de profundidade, com a cor proporcional ao valor de cada pixel (i.e., ao valor de profundidade), útil para efeitos de visualização apenas. O sensor Intel RealSense disponibiliza a sincronização entre câmaras. Ao utilizar a função *wait_for_frames* a aplicação bloqueia até que esteja disponível uma nova imagem de cada uma das várias câmaras. A partir desse ponto, pode obter cada uma das imagens adquiridas, neste caso a de profundidade. A função *colorize* é utilizada aqui para gerar a imagem a cores proporcional à profundidade de cada pixel, com 8 bit por cor, por pixel. A função *get_data*, juntamente com a função *Numpy asanyarray*, é então usada para, com pouca sobrecarga computacional, gerar uma imagem/matriz OpenCV/Numpy que pode ser utilizada com funções OpenCV/Numpy. Se não for especificado nenhum tipo, este é determinado a partir dos dados originais.

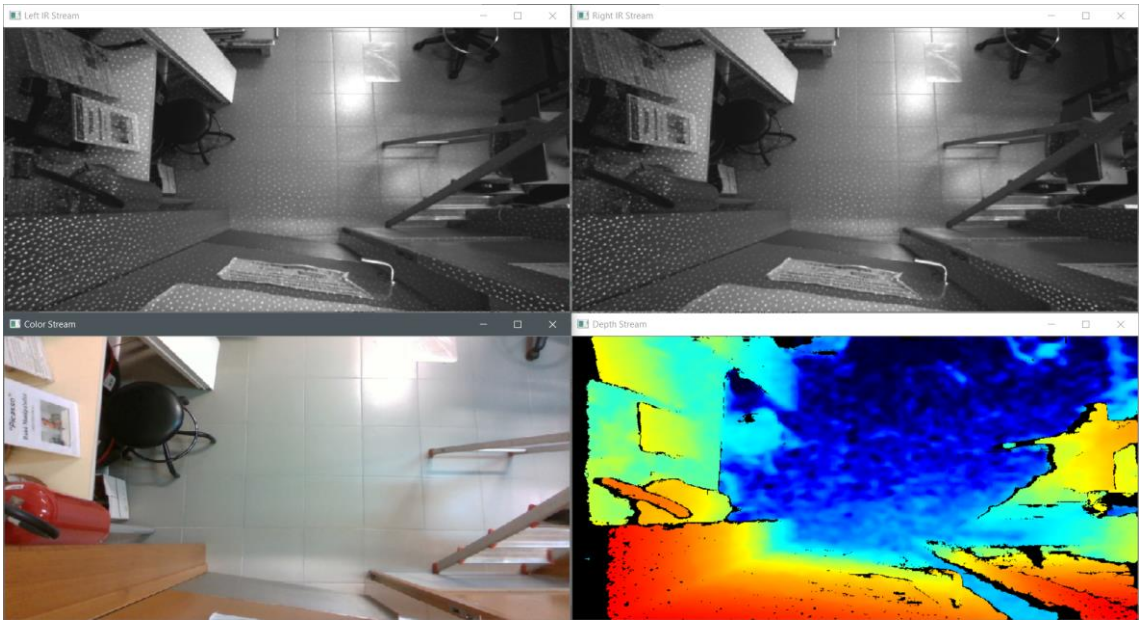

Figura 4 – Vista geral dos fluxos de dados disponíveis no ficheiro bag, das várias câmaras do sensor Intel RealSense D435.

É ainda disponibilizado um outro *script*, com o nome `read_bag_full_example.py`, o qual, além de disponibilizar o mapa de profundidade, mostra ainda os fluxos de dados individuais da câmara infravermelhos e RGB, como ilustrado na Figura 4.

3. Tarefa 1 – Visão estéreo

Embora o sensor Intel RealSense já calcule o mapa de profundidade, nesta tarefa pretende-se avaliar o seu cálculo utilizando os algoritmos lecionados nas aulas.

Os sensores Intel RealSense são calibrados de fábrica [1], podendo as transformações entre cada par de câmaras ser obtido através da função `get_extrinsics_to`. O *script* fornecido

`read_bag_full_example.py` inclui também instruções para obter quer os parâmetros intrínsecos, quer os parâmetros extrínsecos das câmaras, considerando, neste caso, a câmara esquerda como sendo a origem do sensor. No entanto, como estamos a utilizar o ficheiro *bag* para obter estes dados, os valores obtidos poderão não estar corretos. Assim, e como indicado no *script*, iremos simplesmente assumir que todas as câmaras estão paralelas, e que a distância entre as duas câmaras infravermelhos é de 5 cm.

Relativamente aos parâmetros intrínsecos, que podem ser obtidos através do *script* fornecido, e estão também disponíveis nos ficheiros *metadata*, estamos a assumir que não existe distorção em nenhuma das câmaras. Por exemplo, para as câmaras infravermelhos, os parâmetros intrínsecos são dados por (os valores exibidos aqui estão arredondados à 3ª casa decimal):

$$K = \begin{bmatrix} 425.789 & 0 & 426.796 \\ 0 & 425.789 & 234.032 \\ 0 & 0 & 1 \end{bmatrix}$$

Dada a informação acima, **implemente um *script* que calcule o mapa de profundidade a partir dos dados das duas câmaras infravermelhos e compare o seu resultado com a fornecida pelo sensor**. Para converter o mapa de profundidade fornecido pelo sensor em metros, considere a escala utilizada pelo sensor no *script*, em $^{m}/_{pixel\ value}$. A sua implementação deverá correr com o ficheiro *bag* fornecido e mostrada o mapa de profundidade calculado em cada iteração.

Para comprar os seus resultados com o mapa de profundidade calculado pelo sensor, obtenha em cada iteração, e por pixel, a diferença entre o seu mapa de profundidade e o calculado pelo sensor, calculando o valor médio da diferença absoluta e o seu desvio padrão. Caso o valor calculado pelo sensor seja 0, significa que o valor não foi calculado para esse pixel, caso em que pode ignorar o cálculo efetuado para esse pixel (esse pixel não entrará na contabilização realizada para essa iteração). Para evitar problemas de memória no cálculo da média da diferença e o seu desvio padrão ao longo do tempo, utilize o algoritmo Welford's online [2] para cada pixel ao longo do tempo, dado por:

$$\bar{x}_n = \frac{(n-1)\bar{x}_{n-1} + x_n}{n} = \bar{x}_{n-1} + \frac{x_n - \bar{x}_{n-1}}{n}$$

$$\sigma_n^2 = \frac{M_{2,n}}{n},$$

onde

$$M_{2,n} = M_{2,n-1} + (x_n - \bar{x}_{n-1})(x_n - \bar{x}_n),$$

e

n – Número atual total de elementos/imagens, que corresponde também ao instante de tempo discreto de aquisição.
x_n – Elemento calculado no *instante de tempo* n.
$\bar{x}_n$ – Valor médio de todos os n primeiros elementos.
σ_n^2 – Variância dos todos os primeiros n elementos (quadrado do desvio padrão).

Inclua nos seus resultados o tempo médio necessário para calcular cada mapa de profundidade e a taxa média associada. Para medir o tempo necessário no seu *script* pode usar a função `process_time`.

4. Tarefa 2 – Contador de pessoas

Nesta tarefa o seu objetivo é contar o número de pessoas dentro dum laboratório. Para tal, irá necessitar de contar o número de pessoas que entra e sai da sala, considerando que a diferença entre esses dois valores corresponde ao valor que se pretende obter. Pode assumir que inicialmente não se encontra nenhuma pessoa no laboratório.

Para resolver esta tarefa pode utilizar qualquer um dos fluxos de dados (profundidade, infravermelhos e/ou cor). Se usar o mapa de profundidade, pode utilizar o fornecido pelo sensor, ou o calculado na 1ª tarefa deste trabalho. O resultado do algoritmo deverá ser um ficheiro de texto CSV (*Comma Separated Value*, usando a extensão "csv") com o instante de tempo em que a pessoa entrou ou saiu do laboratório (formato HH:MM:SS), o tipo de evento ("in" ou "out) e o número de pessoas que permaneceu no seu interior (número de pessoas que entrou menos o número de pessoas que saiu). A 1ª linha deve indicar o número do grupo, números e nomes de estudante, enquanto a 2ª linha irá indicar o início do programa e número inicial de pessoas, tipicamente 0. A Tabela 1 mostra o exemplo dum ficheiro típico de resultados.

```
# Group X, <num1>, <name1>, <num2>, <name2>
10:22:32,none,0
10:23:32,in,1
10:25:25,in,2
10:27:45,out,1
10:37:48,in,2
10:39:15,out,1
10:57:42,out,0
```

Tabela 1 – Exemplo de um ficheiro de resultados na monitorização do acesso.

5. Créditos extra

Embora a câmara seja calibrada de fábrica, pode efetuar o seu próprio cálculo dos parâmetros intrínsecos e extrínsecos. Os ficheiros de "calibração" fornecidos, onde se visualiza um padrão de xadrez, poderão ser utilizados para esse efeito. Esta tarefa é opcional, mas permiti-lhe obter até 2 valores adicionais. Caso não seja realizada, a nota global do trabalho está limitada a 19 valores. O objetivo é calibrar apenas as câmaras que utiliza.

6. Notas finais

A profundidade calculada a bordo do Intel RealSense, embora utilize o mesmo algoritmo que o discutido nas aulas, inclui ainda um conjunto de pós-processadores para melhorar a estimativa de profundidade. Assim, na Tarefa 1, não espere obter exatamente os mesmos valores que os calculados pelo sensor, não obstante os resultados devam ser próximos.

Podem ser utilizadas diferentes abordagens para a deteção das pessoas a entrar e a sair do laboratório, mas a utilização da profundidade será provavelmente a que dará melhores

resultados e que será menos dependente de alterações do ambiente. Embora seja possível resolver o problema com soluções relativamente simples e rápidas, serão valorizadas soluções que utilizem os algoritmos estudados, tais como os algoritmos de *tracking*.

Embora lhe seja pedido para obter dados temporais relativamente à execução do algoritmo, o fator tempo tem pouca relevância quando comparado com o algoritmo da solução desenvolvida em si. Considere ainda que o código pode ser testado com outros dados, mas sempre assumindo que a câmara será colocada na mesma pose que nos dados fornecidos.

Como indicado no calendário de avaliação, o trabalho deve ser **submetido** no Moodle **até 16 de dezembro**, com as defesas a decorrerem dia 18 de dezembro.

Referências

[1] L. Keselman, J. Iselin Woodfill, A. Grunnet-Jepsen, and A. Bhowmik, 'Intel RealSense Stereoscopic Depth Cameras', Jul. 2017.

[2] B. P. Welford, 'Note on a Method for Calculating Corrected Sums of Squares and Products', *Technometrics*, vol. 4, no. 3, pp. 419–420, 1962, doi: 10.1080/00401706.1962.10490022.