



PROCESSAMENTO DE IMAGEM E VISÃO

MEEC/MEBIOM

Relatório

Autores:

Cátia Fortunato nº 81497
Daniel Fortunato nº 81498
Luís Ferreira nº 84122
Nuno Fernandes nº 81517

:

1º Semestre 2018/2019

1 Introdução

1.1 Definição do Problema

O projeto de Processamento de Imagem e Visão tem com objetivo a detetar, localizar e o rastrear objetos em movimento num ambiente estático. Está dividido em duas partes.

A primeira parte consiste num problema em que é recebido, de uma câmara kinect com posição fixa, uma sequência de imagens cujo o objetivo é detetar os objetos e devolver, para cada um, a sua trajetória através das 8 coordenadas xyz de uma caixa que o contém tal como representado pela fig. 1.

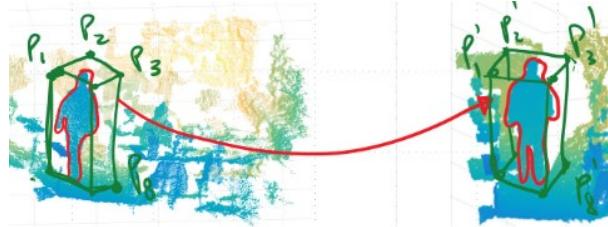


Figura 1: Representação das 8 coordenadas xyz da caixa para cada objeto [1].

A segunda parte consiste no mesmo problema da primeira parte com a diferença de haver duas sequências de imagens recebidas por duas câmara kinect não colineares e estáticas entre si e ainda estáticas em relação ao ambiente. Devido à sua diferente posição no ambiente, as câmaras têm uma visão do mesmo diferente.

1.2 Câmara Kinect e Modelo da Câmera

As câmaras kinect são composta por dois tipos de câmaras, uma com a informação cor e outra com informação da profundidade do ambiente. Com estas duas características, é possível reconstruir o ambiente num imagem a 3 dimensões, denominada de *pointcloud*.

Uma câmara é projeção de pontos em 3D para um plano de imagem. Esta é definida pelo modelo da fig. 2

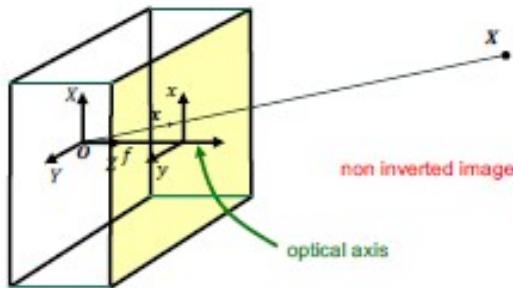


Figura 2: Projeção em perspetiva com plano frontal [2].

na qual os ponto 3D são definidos por $X = [X \ Y \ Z]^T$ e os pontos no plano por $x = [x \ y]^T$ e a relação entre esses pontos em coordenadas cartesianas é

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{Z} \times \begin{bmatrix} X \\ Y \end{bmatrix} \quad (1)$$

Considerando os parâmetros extrínsecos (origem arbitrária do referencial do mundo) e parâmetros intrínsecos (fatores de escala, comprimento focal e ponto principal), em coordenadas homogéneas, o modelo completo da perspetiva da câmara é dado por

$$\lambda \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = P \times \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \quad (2)$$

onde P é uma matriz de 3×4 que contém a informação dos parâmetros e é dada por

$$P = \begin{bmatrix} fs_x & 0 & c_x \\ 0 & fs_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} R & T \\ 0^T & 1 \end{bmatrix}. \quad (3)$$

Na equação 3, a primeira matriz corresponde a parâmetros de conversão de coordenadas métricas para pixel dos parâmetros intrínsecos na qual f é a comprimento focal, c_x e c_y são as coordenadas do ponto principal em pixels e s_x e s_y são fatores de escala em pixel/m. A segunda matriz à passagem de coordenadas cartesianas para coordenadas homogéneas. Por fim, a última corresponde aos parâmetros extrínsecos da câmara, mais especificamente, a matriz de rotação 3×3 entre o referencial do mundo e o referencial da câmara, R , e o vetor de translação 3×1 ou origem do referencial do mundo e o referencial da câmara, T .

Com duas câmaras, tal como no problema 2, facilmente se relaciona o ambiente visto por uma câmara com o ambiente visto pela outra, mas para tal, é necessário a matriz de rotação e vetor de translação entre as câmaras.

Dado um ponto em 3D num referencial de uma câmara 1, esse mesmo ponto no referencial de uma câmara 2 é dado por

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}^{Camara2} = R \times \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}^{Camara1} + T. \quad (4)$$

2 Implementação da Resolução do Problema

2.1 1 Câmera

Para resolução do problema com uma única câmara, começou-se por identificar, a partir da sequência total de imagens, o *background* do ambiente de modo a facilitar a deteção de objetos em movimento. De seguida, removeu-se o *background* de todas as imagens e procedeu-se à deteção de todos os objetos.

Com esta informação, identificou-se para cada imagem e a respetiva imagem seguinte, quais os objetos que são os mesmos, quais são os novos objetos e quais são os que desapareceram das imagens para assim se determinar o trajeto de cada ao longo do ambiente. Este trajeto, tal como referido em 1.1, é composto por 8 coordenadas xyz correspondentes aos vértices de uma caixa que contém cada objeto

De seguida, explica-se detalhadamente as várias fases do problema.

2.1.1 Deteção de Objetos

Em primeiro lugar, com a sequência de imagens *rgb* e as imagens de *depth* (imagem com informação da câmara *depth*), criou-se novas imagens a partir das imagens *rgb* com origem

no referencial da câmara *depth*. tal é feito devido ao facto de se usar a informação da cor como fator de custo na comparação de objetos em 2.1.2.

Para identificar os objetos, em primeiro lugar, é necessário remover o *background* das imagens. Para tal, é aplicado uma mediana às imagens de *depth*. Deste modo, é determinado a imagem mais provável de ser o *background*.

De seguida, para cada imagem, remove-se o *background* e aplica-se à nova imagem um filtro de modo a remover algum ruído criado com estas transformações. Esta última imagem já só contém os objetos que tiveram algum movimento ao longo da sequência. Contudo, é possível que alguns dos objetos sejam de facto um conjunto de vários objetos. Para resolver este problema, aplicou-se a estas imagens, a função de gradiente de modo a identificar o valor da magnitude e direção do gradiente de cada objeto. Assim, identificamos os contornos dos objetos e nos casos em que há objetos compostos por vários, quanto maior for o valor da magnitude, maior será a diferença de profundidade, isto é, a distância entre esses objetos. Assim, com esta informação, é possível separar esses objetos que outrora eram considerados como um só objeto.

Com este problema resolvido, facilmente se deteta e identifica o número objetos em movimento ao longo da sequência de imagens.

2.1.2 Rastreamento de Objetos

Sabendo o número de objetos em cada imagem e o *label* de cada um, procede-se à correspondência dos objetos. Tal é realizado com uma matriz de custo de objetos com tamanho de linhas igual ao número de objetos da imagem *i* e tamanho de colunas igual número de objetos da imagem *i+1*.

Este custo é dividido em 3 componentes: custo de proximidade, custo de volume e de cor.

Para um dado objeto, o custo de proximidade aos outros objetos é determinado pelo cálculo da norma euclidiana do valor dos centroides. Este último valor é calculado a partir da média dos valores dos vértices da caixa.

Como as caixa têm um formato de um paralelepípedo, o custo do volume para cada objeto é determinado através da formula do volume de um paralelepípedo calculado com os valores dos vértices da caixa. Assim, o valor do custo de volume é dado pelo módulo da diferença entre os dois valores.

Quanto o custo da cor, fez-se a comparação entre histogramas de matiz quando a saturação não era baixa e comparação de histogramas de saturação quando esta era baixa.

Deste modo, para cada célula da matriz, o custo relativamente a dois objetos é dado por

$$Custo_{ij} = (CustoProximidade)_{ij} + (CustoVolume)_{ij} + (CustoCor)_{ij}, \quad (5)$$

sendo *i* e *j* as posições da matriz de custo, respetivamente linha e coluna.

Com a matriz totalmente preenchida com os custos calculados por 5, usa-se um algoritmo *greedy* para efetuar a correspondência dos objetos procurando. O algoritmo, em primeiro lugar, determina o(s) valor(es) mínimo(s) de cada linha. Depois, para cada coluna, seleciona o primeiro mínimo entre os mínimos previamente detetados, ou seja, entre vários valores iguais o primeiro é selecionado.

Concluída a correspondência dos objetos, o trajeto de cada objeto ao longo do ambiente é imediatamente adquirido. De notar, que no caso em que o objeto sai e volta a entrar uma *frames* mais tarde este é considerado como um novo objeto.

2.2 2 Câmaras

O problema de 2 câmara tem por base o mesmo tipo de resolução do problema de 1 câmara já que também é necessário identificar objetos em movimento e fazer o respetivo rastreamento ao longo do ambiente.

Contudo, neste problema, para comparar a imagem de cada câmara, isto é, comparar objetos na câmara 1 na câmara 2 e vice versa, é necessário a matriz de rotação e o vetor de translação para se ter uma relação entre pontos das duas câmaras. Neste caso, a matriz e vetor a determinar será a que permite a transformação do referencial da câmara 2 para o referencial da câmara 1.

Para tal, usa-se o *Scale-Invariant Feature Transform* (SIFT) em duas imagens para detetar e corresponder *keypoints*. Aos *keypoints* determinados, aplica-se o *Random Sample Consensus* (RANSAC) e o algoritmo de análise do *Procrustes* para a obtenção da matriz de rotação e vetor de translação.

Com isto, identifica-se os objetos tal com realizado em 2.1.1 para cada câmara e procede-se à fusão das imagens a partir da matriz e do vetor determinado. Só depois é realizado o rastreamento dos objetos.

De seguida, explica-se o funcionamento e a implementação do SIFT, RANSAC, *Procrustes*, da deteção de objetos e por fim o rastreamento de objetos.

2.2.1 SIFT

O SIFT é um algoritmo que extrai *keypoints* de imagens. Começa-se então por determinar os *keypoints* nas imagens da câmara 1 e 2, e procede-se à determinação das *matches* entre estes *keypoints*. As *matches* obtidas neste processo serão utilizadas no passo seguinte para determinar a matriz de rotação e de translação que melhor descrevem estes dados.

Para a determinação dos *keypoints*, calculou-se o *score* do SIFT e foram ordenadas as imagens por ordem crescente de *score*. Finalmente, escolheram-se 10% das images que apresentavam o *score* mais elevado. Os conjunto de pontos destas imagens foi o selecionado para efetuar a estimativa do modelo.

2.2.2 RANSAC

O algoritmo do RANSAC tem com objetivo determinar a melhor aproximação de um determinado modelo para um conjunto de dados, que, neste caso, é a transformação do referencial da câmara 2 para o referencial da câmara 1 através de uma rotação e uma translação. Uma vez que as correspondências que o SIFT calcula não são todas corretas, não é possível fazer uma boa aproximação do modelo usando todos estes pontos. Por essa razão, é usado este algoritmo iterativo que consegue fazer uma melhor aproximação usando apenas dados consistentes.

Com os *keypoints* obtidos pelo SIFT, é possível estimar uma transformação de corpo rígido entre os dois conjuntos de pontos, ou seja, estimar a matriz de rotação R e o vetor de translação T . O RANSAC é um método iterativo que estima os parâmetros do modelo matemático a partir de um subconjunto dos dados e determina o número de *inliers* correspondentes a este modelo. Após várias iterações, devolve o modelo que maximiza o número de *inliers*, isto é, o número de pontos que são corretamente descritos pelo modelo. O RANSAC é composto pelos passos seguintes:

1. Escolher o modelo para ajustar aos dados, que neste caso é dado por $p' = Rp + T$.

2. Selecionar um subconjunto de amostras aleatórias contendo o número mínimo de dados para determinar os parâmetros do modelo a partir do conjunto de dados de entrada. Neste caso, uma vez que pretendemos obter a matriz de rotação R (3x3) e o vetor de translação T (3x1) são precisos, no mínimo e apenas, 4 pontos. Desta forma é garantida a remoção de todos os *outliers*.
3. Ajustar o modelo usando apenas o subconjunto selecionado anteriormente. Para este caso, R e T são calculados, para todas as iterações, através do *Procrustes* (ver 2.2.3).
4. Verificar que elementos do conjunto total de dados são consistentes com os parâmetros calculados. Os elementos são considerados como *outliers* se não se ajustarem ao modelo calculado apenas com a diferença de um erro (usou-se o *mean square error*) que, neste caso, o desvio máximo do ruído é definido por um *threshold*.
5. Repetir os passos 2, 3 e 4 para o número de iterações escolhido.
6. Utilizar os dados que geram o modelo que cria o maior número de *inliers* para calcular a matriz de rotação e translação que melhor descrevem os dados utilizados.

Para se perceber melhor a implementação deste algoritmo para este projeto, está em baixo a figura 3 que explica de uma forma mais resumida a abordagem utilizada do cálculo deste modelo.

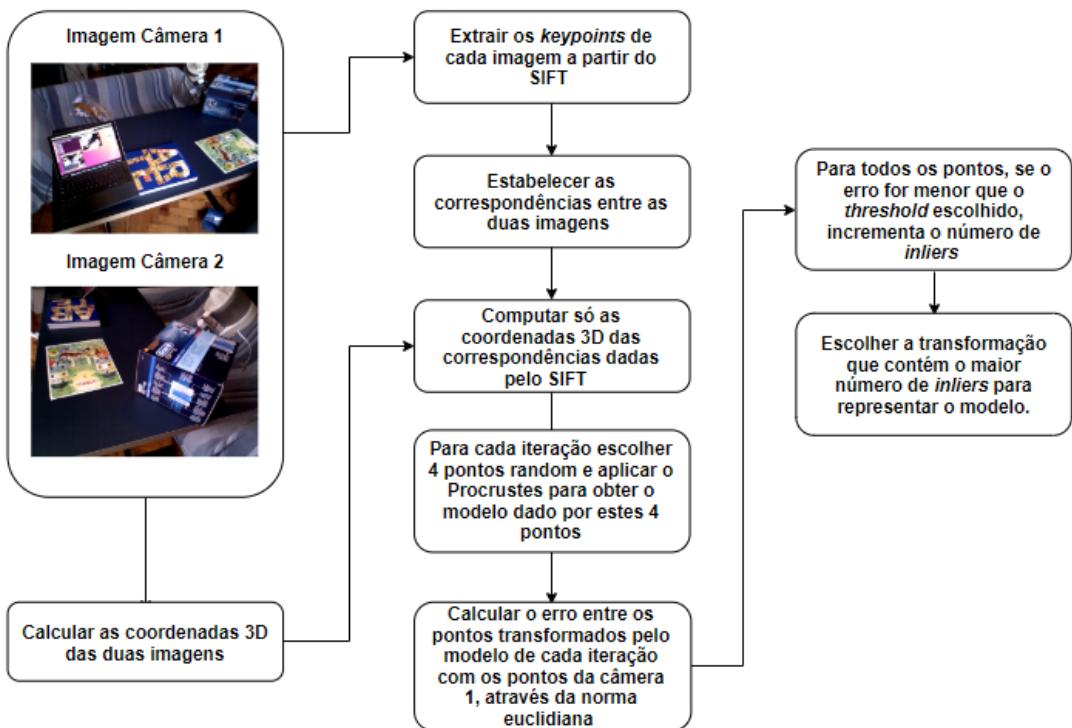


Figura 3: Diagrama do algoritmo RANSAC aplicado a este projeto

2.2.3 Procrustes

O método do *Procrustes*, referido anteriormente na implementação do RANSAC, é um método utilizado para calcular a matriz de rotação ideal entre dois pontos. O algoritmo do *Procrustes* assume que existe dois conjuntos de pontos $\{p_{1i}\}$ e $\{p_{2i}\}$ relacionados através da equação

$$p'_i = Rp_i + T \quad (6)$$

onde R é a matriz de rotação (3×3) e T é o vetor de translação 3D.

Na procura da melhor transformação para representação do modelo em estudo, usou-se este algoritmo duas vezes: uma primeira, dentro do algoritmo do RANSAC, para calcular a transformação dos pontos aleatórios das correspondências dadas pelo SIFT, e uma segunda para calcular a transformação que contém o maior número de *inliers*, dado mais uma vez pelo RANSAC, para que esta transformação seja a que representa o modelo final entre as duas câmeras.

2.2.4 Deteção dos Objetos

Tal como para o problema de 1 câmara, utiliza-se remoção do *background* e a função do gradiente nas imagens de *depth* para cada uma das câmera, detetando assim objetos em movimento em dois referenciais de coordenadas (referencial da câmara 1 e referencial da câmara 2). Aplica-se a transformação calculada como descrito em 2.2 aos objetos da câmara 2 de modo a obter as coordenadas com origem no referencial do mundo (câmara 1). Por fim, procede-se à correspondência de objetos com base num custo de proximidade entre eles. Aqueles que são identificados enquanto o mesmo objeto detetados nas duas câmaras têm uma fusão de informação. Tudo o resto é mantido enquanto objeto independente já em coordenadas do referencial do mundo.

2.2.5 Rastreamento de Objetos

À semelhança da situação com uma câmara, visto que a informação dos objetos foi fundida na deteção/extração dos objetos, o rastreamento é feito através da escolha de um menor custo entre dois objetos de duas imagens consecutivas, superior a um certo limite.

3 Resultados Experimentais

3.1 1 Câmara

Testou-se a implementação da resolução do problema 1 com os conjuntos de dados "um", "bonecos" e "filinha".

3.1.1 Conjunto de dados "um"

Neste conjunto de dados existem trinta e oito imagens. Como existe pouco movimento, o *background* foi feito com sucesso permitindo uma extração facilitada dos objetos. Sendo que estes objetos eram de dimensões superiores às consideradas de erro de leitura foram identificados corretamente os objetos e posteriormente o caminho foi bem definido.

Tabela 1: Objetos identificados conjunto de dados "um"

Objeto	Imagens detetado	Número de deteções
1	[1 ... 38]	38
2	[34 ... 38]	5

Como se vê na tabela 1 apenas foram detetados dois objetos.

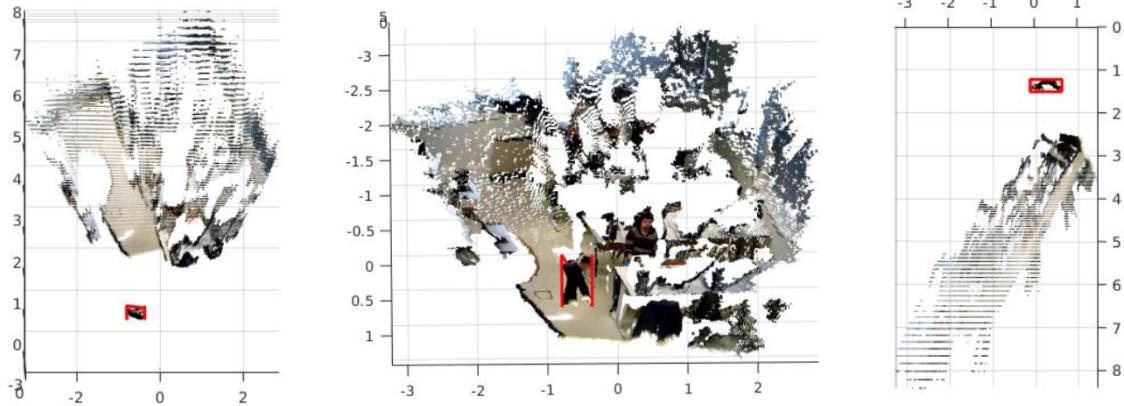


Figura 4: Deteção do objeto 1, em 3 perspetivas

A caixa engloba a *point cloud* do objeto detetado com os pontos máximos em cada eixo do referencial mundo.

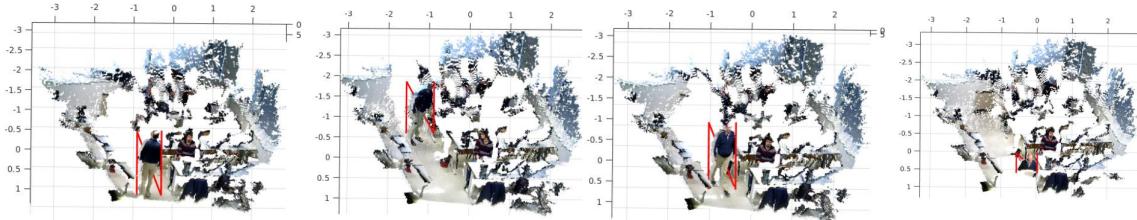


Figura 5: Deteção do objeto 1, movimento em 4 imagens

O objeto em movimento é detetado enquanto o mesmo objeto de imagem para imagem, ilustrado com quatro das imagens retiradas.

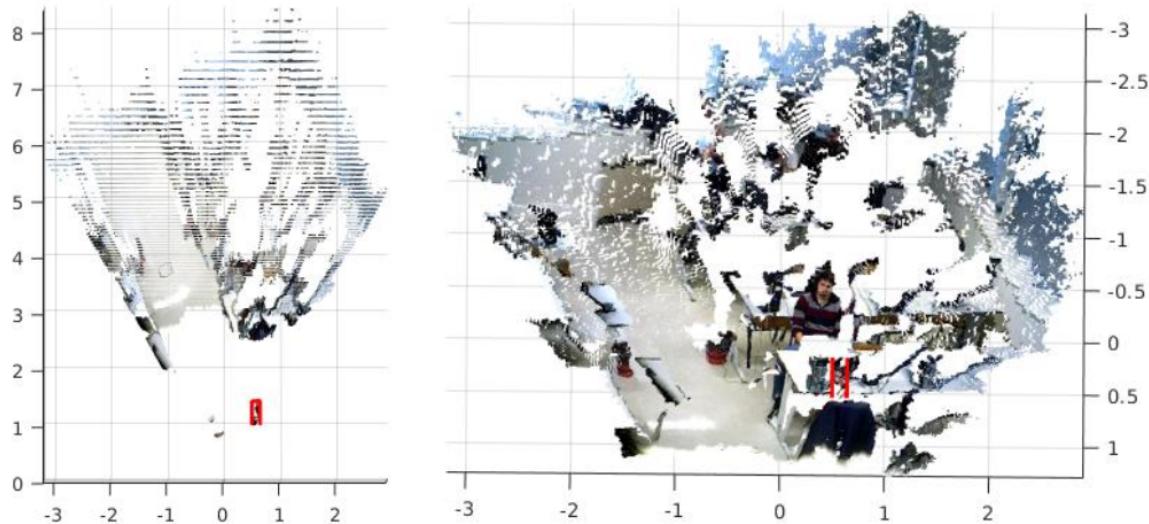


Figura 6: Deteção do objeto 2, em 2 perspectivas

O segundo objeto detetado é possível de identificar uma cabeça a aparecer nas últimas imagens.

3.1.2 Conjunto de dados "bonecos"

Neste conjunto foram detetados dezanove objetos. No entanto, este valor não está correto. A justificação para tal inclui um *background* mal detetado e que objetos sobrepostos e próximos (que resulta num gradiente baixo) são detetados como um só.

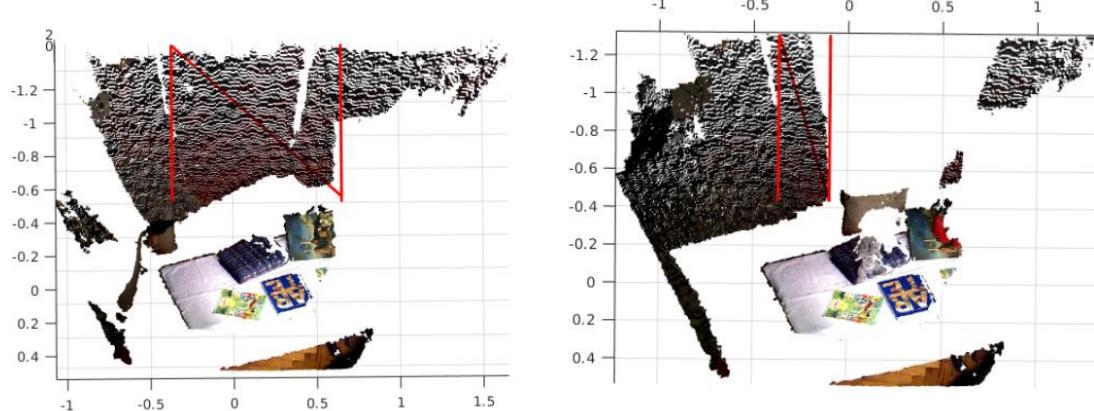


Figura 7: Deteção do fundo enquanto objeto, 2 ocasiões

Como a mediana de todos as imagens inclui a existência de um objeto, o fundo é detetado enquanto objeto até a pessoa se movimentar para o local em que a mediana foi feita como visto na figura 7.

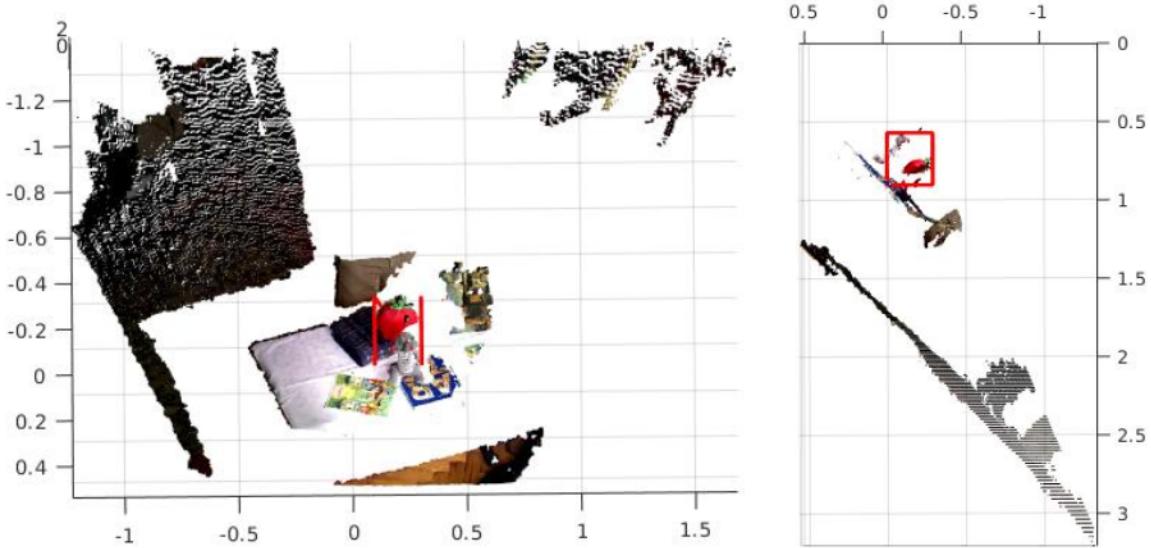


Figura 8: Deteção de dois objetos enquanto um, em 2 perspetivas

Sendo os objetos de dimensões reduzidas, o filtro de gradiente que se utilizou para separar os objetos sobrepostos não foi suficiente resultando na junção de dois objetos como um só como visto na figura 8.

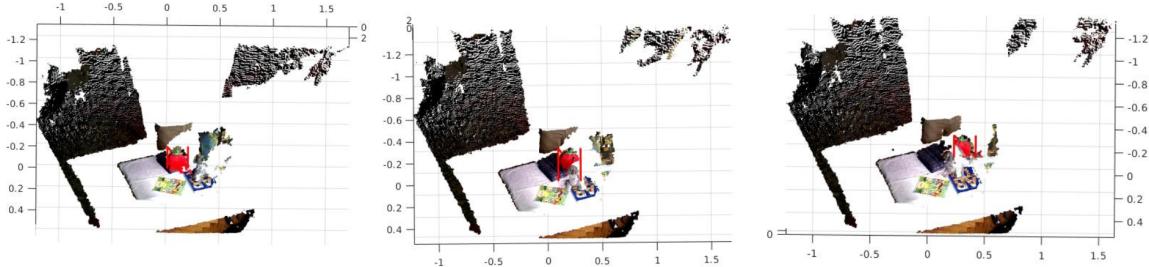


Figura 9: Deteção de um objeto enquanto três objetos diferentes.

O mesmo objeto está nas três imagens, no entanto a imagem central da figura 9 está na situação da figura 8, desta forma o objeto entre imagens altera drasticamente o seu volume e cor resultando na distinção de um novo objeto que separa o caminho, neste troço, do objeto inicial (o boneco vermelho).

Outras fontes de erro para o algoritmo neste conjunto de dados advêm do ajuste de constantes de limite para a extração de objetos, de forma a reduzir o ruído ao máximo incluindo os objetos, e para o rastreamento.

3.1.3 Conjunto de dados "filinha"

Neste conjunto foram detetados vinte e três objetos. No entanto, este valor não está correto. Justifica-se por incluir um *background* mal detetado.

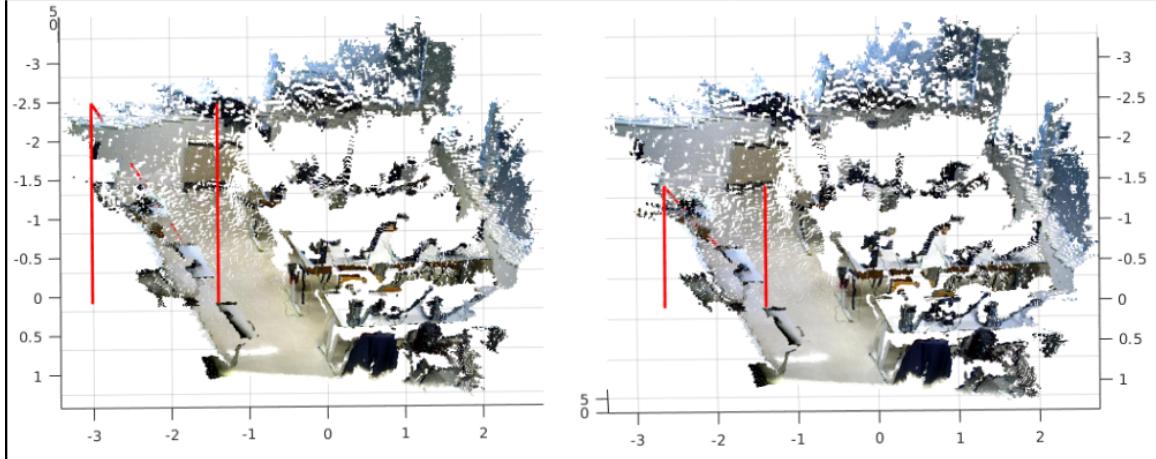


Figura 10: Deteção do plano de fundo enquanto 2 objetos.

Visto que o próprio do plano de fundo tem um gradiente elevado, este é dividido em vários objetos enquanto não é ocupado como visto na figura 10, e ignora a ocupação quando tem o valor que corresponde ao plano de fundo (um objeto). Isto também afeta o rastreamento com uma quebra no caminho do objeto, que de uma imagem para outra passa a ser plano de fundo.

3.2 2 Câmara

Testou-se a implementação da resolução do problema 2 com os conjuntos de dados "fruta1", obteve-se insucesso e não foram testados mais resultados.

Este insucesso deve-se ao facto de a determinação da matriz de rotação e do vetor de translação não estar correta. Por isso, algum problema que não se encontrou está na parte da implementação do SIFT, RANSAC e *Procrustes*. Contudo, conclui-se que o erro não está na parte do RANSAC uma vez que testou-se adicionando *outliers* e *noise* e, nesta parte, a matriz e o vetor permaneceram aproximadamente iguais.

Criou-se uma matriz de rotação dada por

$$R_{fabricado} = \begin{bmatrix} 0,3106172175 & 0,6463858629 & 0,6969234251 \\ -0,8047378541 & 0,5690355937 & -0,1691019787 \\ -0,5058793634 & -0,5083146755 & 0,6969234251 \end{bmatrix}.$$

Adicionando várias percentagens de *outliers* com um intervalo de [0m; 2m] como 40% e 80%, pelo RANSAC e de seguida o procrustus, obtém-se as seguintes matrizes de rotação

$$R_{40\%} = \begin{bmatrix} 0,3106172175 & -0,8047378541 & -0,5058793634 \\ -0,6463858629 & 0,5690355937 & -0,5083146755 \\ 0,6969234251 & -0,1691019787 & 0,6969234251 \end{bmatrix}.$$

$$R_{90\%} = \begin{bmatrix} 0,9589716475 & -0,2107067898 & -0,1896734775 \\ 0,2602107522 & 0,9197478789 & 0,2938608576 \\ 0,1125333007 & -0,331159309 & 0,936840311 \end{bmatrix}.$$

Só para uma percentagem de 80% é que existem variações em relação a $R_{fabricada}$

Os resultados com a implementação acima mencionada para a transformação rígida não são satisfatórios, na figura ?? consegue-se ver após a fusão de duas imagens dois planos correspondentes à mesa, consequentemente a identificação e o rastreamento de objetos iria ter por base um erro que impedia o correto funcionamento do programa.

4 Conclusão

Verificou-se que uma passo muito importante para a deteção de objetos em movimento é o de detetar com um erro reduzido a imagem do *background*. Isto porque em zonas de elevado movimento, o *background* não é bem definido e por isso aquando do processo de deteção de objetos, por vezes, o próprio *background* nessa zona é considerado como um objeto tal como aconteceu no problema 1 no conjunto de dados "bonecos" e "filinha". Por isso, para detetar o *background* sugere-se outro processo diferente do usado, como por exemplo *multi-colour background model per pixel* proposto por Grimson [3] ou então, para o caso de o ambiente envolver sombras, o *Improved Adaptive Background Mixture Model for Realtime Tracking with Shadow Detection* [4] isto porque o primeiro referido não distingue entre objetos em movimento e sombras em movimento.

Também se verificou que desenvolver um algoritmo geral de deteção e rastreamento de objetos é complicado já que à partida não se sabe nenhuma informação do objeto como por exemplo tamanho, volume, entre outros. Assim, na etapa de reduzir o ruído e detetar objetos, pode acontecer, por exemplo, estar-se a desprezar certos objetos com um tamanho pequeno.

Referências

- [1] <http://printart.isr.ist.utl.pt/piv/project/>
- [2] Marques, J., Santos-Victor, J., 2018, Camera model, Instituto Superior Técnico, 14 pp.
- [3] http://www.ai.mit.edu/projects/vsam/Publications/stauffer_cvpr98_track.pdf
- [4] <http://www.ee.surrey.ac.uk/CVSSP/Publications/papers/KaewTraKulPong-AVBS01.pdf>