Instituto Superior Técnico

Processamento de Imagem e Visão

MEEC 2018-2019 - 1º Semestre

Deteção e localização de objetos através de múltiplas câmaras de profundidade

Autores:
André Meneses, 84005
João Sebastião, 84087
John Mendonça, 84106
Rúben Gomes, 84180

Docente:
João P. Costeira

Conteúdo

1 Introdução

O objetivo deste projeto é a deteção, localização e rastreamento de objetos em movimento através da informação de câmaras de profundidade. As câmaras estão fixas e a cena é composta por objetos estáticos (fundo) e um número finito e variável de objetos em movimento. Para a realização do trabalho, utilizaram-se conjuntos de dados obtidos com sensores Kinect. Um sensor Kinect é composto por dois tipos diferentes de câmaras: uma câmara que nos dá informação sobre a profundidade da cena e uma câmara que nos dá informação sobre a cor. Um exemplo deste sensor está representado na figura 1.

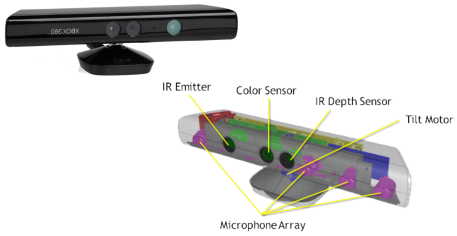


Figura 1: Sensor kinect

Estas câmaras fornecem dois tipos de imagem - imagens de cor (RGB) e profundidade - com a mesma resolução. Ao combinar a informação proveniente das duas imagens, é possível reconstruir a cena em 3D, atribuindo a cor dos píxeis ao ponto 3D correspondente, obtido com a câmara de profundidade. Na figura 2 encontram-se exemplos dos dois tipos de imagens.

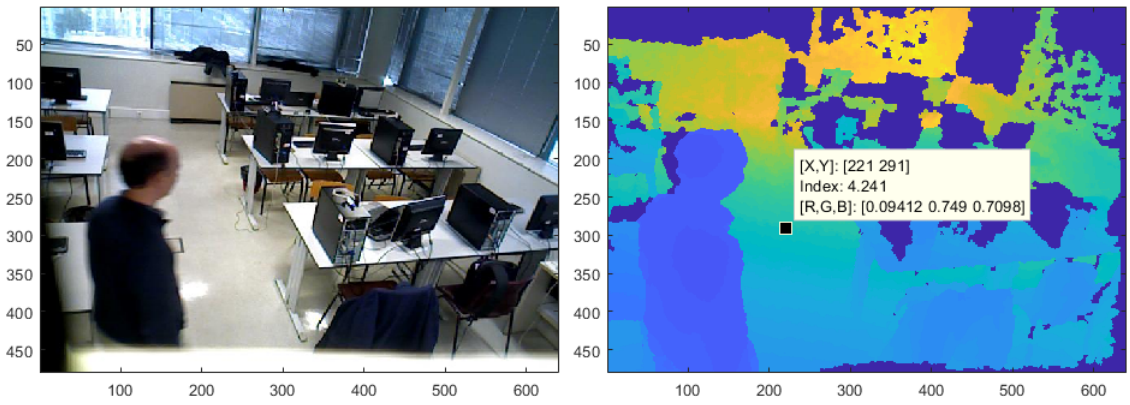


Figura 2: Exemplos de imagens rgb (esquerda) e de profundidade(direita)

2 Definição do problema

O projeto é constituído por 2 partes, cujos objetivos se descrevem de seguida:

2.1 Parte 1 - Uma câmara

A primeira parte do projeto consiste em detetar, localizar e rastrear objetos em movimento, através de uma sequência de imagens de profundidade e de cor obtidas com um sensor Kinect estático. Para cada objeto detetado numa imagem, retornam-se as 8 coordenadas xyz, correspondentes aos vértices de uma caixa que engloba o objeto nessa imagem.

2.2 Parte 2 - Duas câmaras

A segunda parte tem o mesmo objetivo que a primeira, mas neste caso, através de uma sequência de imagens de profundidade e de cor obtidas por dois sensores Kinect estáticos. A posição relativa entre os dois sensores não é alterada. Neste caso, além das 8 coordenadas retornadas para cada objetado detetado em cada imagem, retornam-se também as matrizes de rotação e de translação entre a câmara 2 e a câmara 1.

3 Abordagem do problema

Tanto para a parte 1, como para a parte 2, dividiu-se o problema em sub-problemas. Nas secções seguintes, para cada uma das partes, descreve-se em termos gerais a solução adotada para o problema, seguida de uma explicação pormenorizada de cada sub-problema.

3.1 Parte 1

Para cumprir os objetivos propostos na parte 1, começou-se por obter uma função que dado o valor do píxel (u, v) obtém-se o valor da cor rgb e das coordenadas xyz correspondentes. Este processo foi realizado através do modelo da câmara, sendo os parâmetros das câmaras fornecidos. O passo seguinte consiste na deteção e identificação de objetos em movimento. Para tal, calcula-se o *background* através de uma mediana, detetam-se os objetos que não pertencem ao *background* e resolve-se um problema de conectividade para diferenciar os objetos. De seguida, com informação das coordenadas xyz de cada píxel e tendo cada objeto identificado por etiquetas, calcula-se para cada um as coordenadas dos vértices da caixa que os engloba. Por fim, resolve-se o problema de correspondência de objetos entre imagens diferentes através do cálculo de uma função com 2 parcelas referentes à cor e à distância dos centros de massa.

3.1.1 Obtenção da cor e das coordenadas xyz para cada pixel em uma imagem

Dadas as imagens de profundidade e rgb, consegue-se obter uma função que mapeia as coordenadas da imagem rgb, [u v], nos valores da cor rgb e das 3 coordenadas xyz. Para resolver este problema, recorrendo ao modelo da câmara , é necessário conhecer os parâmetros intrínsecos e extrínsecos de cada câmara. O modelo da câmara pode ser expresso, em coordenadas homogéneas, por

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} K \end{bmatrix} \begin{bmatrix} R & T \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \tag{1}$$

onde a matriz K é dada por

$$\begin{bmatrix} fs_x & 0 & c_x \\ 0 & fs_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

e representa os parâmetros intrínsecos da câmara. A matriz [R|T] é uma matriz 3×4 e representa os parâmetros extrínsecos da câmara. A resolução deste problema assenta em 3 etapas, descritas de seguida:

- Obtenção dos pontos 3D no sistema de coordenadas da câmara de profundidade

- Transformação dos pontos 3D no sistema de coordenadas da câmara de profundidade para o sistema de coordenadas da câmara rgb com matrizes R e T conhecidas
- Estabelecimento das coordenadas dos pontos 3D no sistema de coordenadas da câmara rgb com os píxeis da imagem rgb.

Os pontos (x, y, z) nas coordenadas da câmara de profundidade são obtidos, assumindo que o sistema de coordenadas do mundo e da câmara de profundidade são coincidentes, pelo que a matriz [R|T] é a matriz identidade com uma coluna de zeros à direita. Invertendo o sistema vem que:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}^{depth} = \begin{bmatrix} \ K \ \end{bmatrix}^{-1} \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{3}$$

Uma vez que as matrizes de rotação e translação entre a câmara de profundidade e de cor são conhecidas, consegue-se obter os pontos (x, y, z) nas coordenadas da câmara rgb.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}^{rgb} = \begin{bmatrix} \ R \ \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}^{depth} + \begin{bmatrix} \ T \ \end{bmatrix} \tag{4}$$

Por fim, estabelece-se a ligação entre os pontos em 3D nas coordenadas da câmara rgb e os píxeis da imagem rgb, aplicando diretamente a expressão (1), utilizando os parâmetros intrínsecos e extrínsecos da câmara rgb.

3.1.2 Deteção e identificação de objetos em movimento

A cena é composta por objetos estáticos (fundo) e um número de objetos em movimento. Para detetar os objetos em movimento começa-se por calcular o fundo. Para calcular o fundo, a partir de n imagens obtidas pela câmara de profundidade, ao longo do tempo, toma-se como valor de cada píxel da imagem de fundo a mediana dos mesmos píxeis de todas as imagens, ou seja, dadas as coordenadas de um píxel (u,v) tem-se:

$$I(u,v)^{fundo} = mediana(\{I_1(u,v), I_2(u,v), ..., I_n(u,v)\}) \tag{5}$$

Tendo a imagem de fundo calculada, a deteção de objetos em cada imagem, $I(u,v)$, é realizada computando a diferença, em valor absoluto, da imagem pela imagem de fundo (píxel por píxel), obtendo-se:

$$I(u,v)^{obj} = |I(u,v) - I(u,v)^{fundo}| \tag{6}$$

Para cada píxel de $I(u,v)^{obj}$, verifica-se se o valor é maior do que um determinado *threshold*, obtendo-se uma imagem binária, $I(u,v)^{bin}$, que em cada píxel toma o valor de 1 caso a verificação anterior seja afirmativa e 0 caso contrário. Existem imperfeições na obtenção do valor da coordenada z por parte do Kinect, uma vez que este não atua corretamente em superfícies pretas e devido ao ruído que se estabelece nos contornos dos objetos. Deste modo, aplica-se à imagem binária um filtro morfológico (não linear) de erosão, seguido de uma dilatação, eliminando ruído que possa existir entre dois objetos próximos, como se representa na figura 3.

Após a eliminação do ruído tem-se uma imagem binária que representam os píxeis onde são detetatos objetos. No entanto, dois píxeis com o valor de 1 não correspondem necessariamente

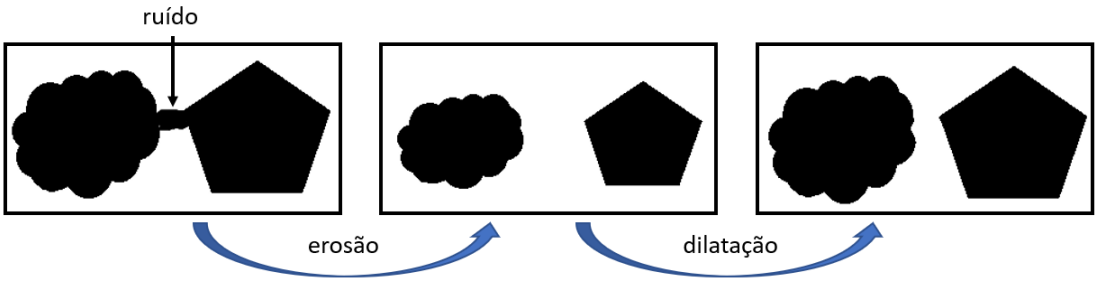


Figura 3: Processo de filtragem morfológica

ao mesmo objeto. Para fazer esta divisão, através do gradiente da coordenada z, calculam-se os contornos dos objetos e substitui-se os valores desses píxeis de 1 para 0 tendo, desta forma, uma imagem binária onde cada objeto está dividido de outro por zeros, como se demonstra na figura 4.

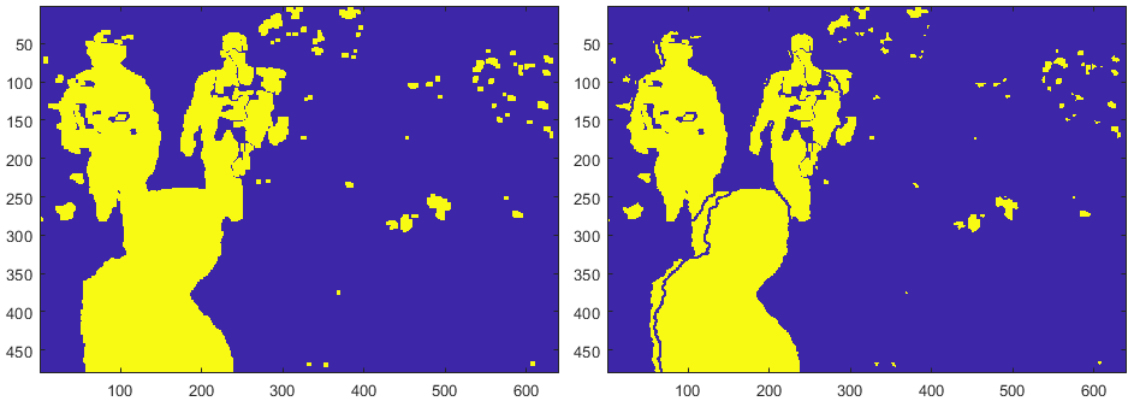


Figura 4: Imagem binária antes e após o cálculo dos contornos e divisão dos objetos

Por fim, etiquetam-se os objetos considerando que um píxel pertence ao mesmo grupo de outro se for vizinho, onde o critério de vizinhança é o de 8 píxeis. Na figura 5 apresentam-se os 8 píxeis vizinhos (verde) de um píxel (azul) de acordo com este critério:

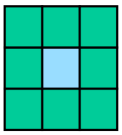


Figura 5: Critério de vizinhança de 8 píxeis

A figura 6 mostra que as 3 pessoas detetadas pela câmara são identificadas separadamente, como se pode observar pelo identificador *index*, tal como esperado. O ruído presente na imagem será posteriormente eliminado.

3.1.3 Determinação das coordenadas dos vértices da caixa que engloba um objeto

Tendo a imagem etiquetada, em que cada objeto tem um índice identificador, para cada objeto procura-se e guardam-se as linhas e colunas (píxeis) onde o mesmo aparece. Se o número de píxeis for menor do que um *threshold*, que se considerou 700, ignora-se, pelo que não são obtidas

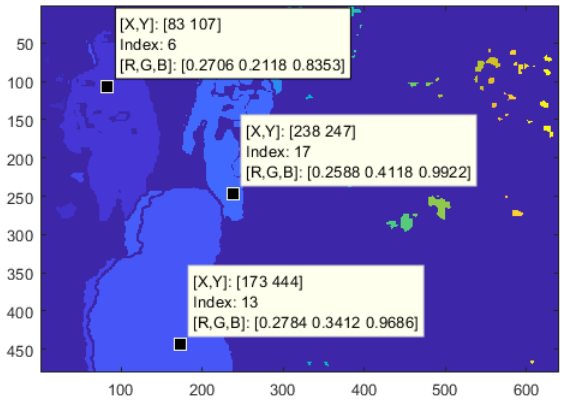


Figura 6: Label dos objetos detetados pela câmara

caixas para objetos de pequenas dimensões nem ruído que ocupe poucos píxeis. Caso ocupe um número superior a 700 píxeis, percorre-se todos os píxeis e caso a coordenada z de cada píxel for diferente de 0 (superfície negra - erro), analisam-se as coordenadas x, y e z e, se necessário, atualizam-se os valores máximos e mínimos de cada coordenada. No final de percorrer os píxeis desse objeto, tem-se os valores máximos e mínimos de cada coordenada que correspondem aos 6 planos que formam a caixa que engloba o objeto e as suas interseções dão origem aos 8 vértices. A figura 7 mostra a *pointcloud* correspondente à imagem apresentada na figura 6.

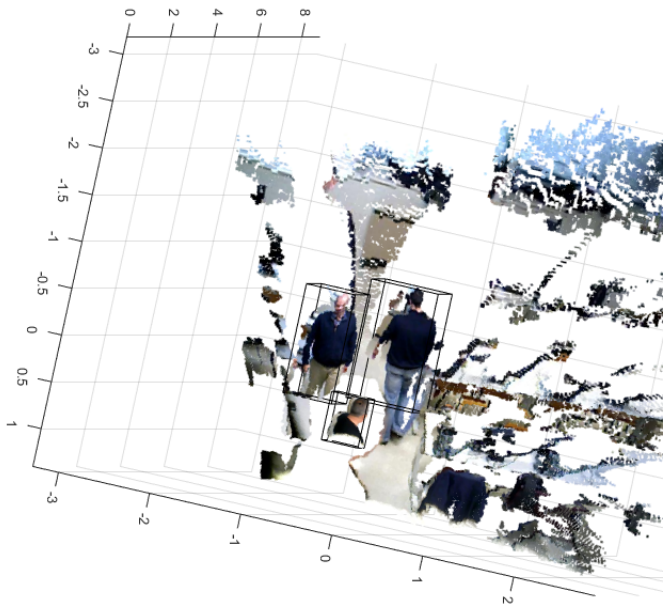


Figura 7: Pointcloud obtida e desenho das caixas

Como se verifica, foram desenhadas 3 caixas que englobam as 3 pessoas presentes na imagem. O ruído presente na figura 6 não foi contabilizado no desenho das caixas, uma vez que não ocupa um valor superior a 700 píxeis, tal como pretendido.

3.1.4 Correspondência entre objetos presentes em imagens diferentes

Um requisito deste projeto é fazer uma correspondência entre o mesmo objeto que apareça em imagens distintas, que foram obtidas pela câmara em dois instantes de tempo diferentes. Para este efeito, usaram-se duas caraterísticas dos objetos de modo a poder fazer a comparação. As caraterísticas são a cor e a distância, que se explicam em detalhe de seguida.

Cor

Em cada imagem, tem-se a informação da cor rgb de cada píxel, no entanto, fazer uma comparação entre objetos através desta informação não daria bons resultados, uma vez que se em imagens distintas o mesmo objeto aparecesse numa zona com menos ou mais sombra a cor rgb iria alterar-se, embora a cor do objeto seja a mesma. Para contornar este problema, usando a informação da cor rgb ({R, G, B}), normaliza-se a cor de tal modo que

$$r = \frac{R}{R+G+B};\;\; g = \frac{G}{R+G+B};\;\; b = \frac{B}{R+G+B}, \tag{7}$$

tendo deste modo uma cor rgb normalizada ({r, g, b}). Esta cor é definida por um plano dado pela equação

$$r + g + b = 1. \tag{8}$$

Assumindo o centro do plano o ponto $(1/3, 1/3, 1/3)$, que representa a cor branca, pode-se trabalhar em coordenadas polares onde a distância ao centro corresponde à saturação (indicação da quantidade de branco) e o ângulo corresponde à tonalidade principal da cor. Isto corresponde à transformação da cor rgb para hsv. Assim sendo, a variável mais importante na comparação de cores entre dois objetos é a tonalidade principal, tendo a saturação maior importância quando se está perante objetos com a mesma tonalidade principal (*hue*) mas com diferente intensidade de cor (*saturation*).

A comparação da cor entre objetos é feita à custa do cálculo da similaridade entre histogramas de *hue* e *saturation*, através do coeficiente de *Bhattacharyya* [1]:

$$D(p,q) = 1 - BC = 1 - \sum_{x \in X} \sqrt{p(x)q(x)} \tag{9}$$

em que p e q representam dois histogramas normalizados de *hue* e *saturation* para cada imagem. Quando comparados dois objetos iguais, idealmente tem-se que $D = 0$.

Distância

Espera-se que um objeto não percorra uma grande distância em duas imagens obtidas pela câmara consecutivamente, ou seja, é de esperar que a distância entre os dois pontos de posição (em 3D) não tome um valor alto. Para cada objeto calcula-se o ponto central da caixa, $(x, y, z)^{central}$, que o engloba, que é dado por

$$cm = (x, y, z)^{central} = \Big(\frac{x_{max} + x_{min}}{2}, \frac{y_{max} + y_{min}}{2}, \frac{z_{max} + z_{min}}{2}\Big). \tag{10}$$

Comparando dois objetos de imagens diferentes, considera-se que a distância entre eles é dada pela norma do vetor cujas extremidades são os pontos centrais de cada um dos objetos, ou seja,

a distância entre os dois centros de massa A e B, correspondem à distância euclideana

$$d_{cm_A,cm_B} = ||\overrightarrow{cm_A} - \overrightarrow{cm_B}|| \tag{11}$$

Função de custo

A função de custo utilizada para detetar a presença de um dado objeto em movimento em duas imagens diferentes é a soma de duas parcelas, uma referente à cor e outra referente à distância. A parcela referente à cor toma valores entre 0 e 1. Por este motivo, normaliza-se a parcela da distância dividindo a distância entre os dois centros de massa pelo valor máximo de uma coordenada obtido no *background*, de forma a não obter um peso muito maior para a distância do que para a cor. Esta normalização não implica que a parcela referente à distância entre dois centros de massa tome valores apenas entre 0 e 1, mas num caso normal, como por exemplo no *dataset* da sala de aula em que se obtém um valor máximo de z de 8 metros, esta normalização garante que dois objetos em imagens consecutivas tenham uma distância normalizada entre 0 e 1. Obtém-se a seguinte expressão da função de custo em função da distância de cores, *cor*, e da distância entre os dois centros de massa, *dist*, apresentadas anteriormente:

$$f(cor, dist) = cor + k \cdot \frac{dist}{z_{max}}, \tag{12}$$

onde k é um hiperparâmetro ajustável de forma a dar o ênfase pretendido à distância entre os centros de massa. Para cada objeto, presente na *frame* atual, calcula-se a função de custo relativa aos objetos da *frame* anterior e o que apresentar função de custo mínima, se estiver dentro de um *threshold*, faz-se a correspondência entre os objetos.

3.2 Parte 2

Para cumprir os objetivos propostos na parte 2, aplica-se todo o conhecimento e métodos obtidos na parte 1, mas introduzem-se métodos relacionados com a correspondência da informação obtida por cada uma das câmaras. A obtenção das matrizes R e T é feita à custa dos métodos *Procrustes* e *RANSAC* que tiram partido de todos os *inliers* para as obter. Estes processos serão explicados em detalhe nas secções seguintes.

3.2.1 Obtenção das matrizes de transformação R e T entre as duas câmaras

A obtenção da matriz de rotação R e do vetor de translação T, que mapeia um ponto 3D do sistema de coordenadas da câmara 2 para o sistema de coordenadas do mundo (câmara 1) é um problema conhecido de Processamento de Imagens, e é designado por *Procrustes Analysis*. Este método permite determinar uma transformação linear entre dois conjuntos de pontos X e Y, correspondentes entre si, através da minimização do critério da soma dos erros quadráticos:

$$R^*, T^* = \arg\min_{R,T} \sum_{i=1}^{K} \|Y - RX - \mathbf{T}\|^2 \tag{13}$$

em que R corresponde à matriz de rotação com dimensão 3×3 e $\mathbf{T}$ corresponde à componente de translação com dimensão 3×1. Desta forma, é possível relacionar os pontos X com os pontos Y através da seguinte relação:

$$Y = RX + \mathbf{T} \tag{14}$$

Note-se que esta relação só é válida quando o critério de correspondência é mínimo, ou seja, quando o matching é perfeito. Na maioria dos casos isto não é possível, pelo que existe uma componente de erro aditivo na equação (14).

A obtenção do conjunto de pontos X e Y, correspondentes, é realizada através de um processo que envolve a obtenção de pontos chave de cada imagem, e posterior correspondência entre eles. Os pontos chaves são conjuntos de píxeis de uma imagem que contém informações relevantes para processamento, nomeadamente reconstrução 3D. Estes pontos chave recolhem partes de uma imagem nomeadamente esquinas, contornos (associados a variações bruscas de intensidade - gradiente com valores elevados, preferencialmente em várias direções) ou cor (associado a histogramas de cor/intensidade), que contêm características que os diferenciam dos restantes pontos da imagem. Estas características são invariantes em relação à rotação e escalamento e, portanto, são utilizados neste tipo de problemas, em que é necessário encontrar correspondência entre pontos de imagens diferentes. A obtenção deste pontos é feita à custa do algoritmo *SIFT*, *Scale-Invariant Feature Transform*[2]. Este algoritmo deteta os pontos chave e os seus *descriptors*, que consistem num vetor de *features*. O detetor extrai da imagem regiões que são consistentes a variações de condições de ponto de vista, nomeadamente iluminação. O *descriptor* associa a cada região uma assinatura que identifica a sua aparência de maneira compacta e robusta.

Cada ponto chave é definido como um disco com centro, escala e orientação. Os *descriptors* correspondem a um vetor que contém os valores de todas as entradas de histogramas de orientação. No nosso caso é utilizado um vetor característica com 128 elementos que corresponde a 16 histogramas para cada *descriptor* multiplicado por 8 orientações, cujos resultados foram demonstrados como os melhores segundo [2].

A correspondência entre features de imagens diferentes é feita utilizando a distância euclidiana entre os vetores do *descriptor*, com a adição de um *threshold* variável chamado *distance ratio* que exclui correspondências cujas distâncias $d(D1, D2)$ entre uma correspondência de vetores candidata multiplicada pelo *distance ratio* retorne um valor inferior à distância entre $D1$ e todos os *descriptors*.

A correspondência que é feita entre os pontos chave de duas imagens diferentes apresenta correspondências incorretas, como se apresenta na figura 8. Para remover correspondências incorretas utiliza-se o método *RANSAC* [3], responsável por remover *outliers* de um conjunto de dados. Consiste num processo iterativo e em cada ciclo realizam-se 4 etapas:

1. Escolher aleatoriamente, do conjunto de pontos chave, o número mínimo de pontos para instanciar o modelo de rotação e translação (4 pontos), no nosso caso.

2. Classificar os restantes pontos como *inlier* ou *outlier*: aplica-se a cada ponto chave de uma imagem a transformação rígida com as matrizes R e T obtidas no passo anterior, e determina-se a distância euclideana entre o ponto obtido com a transformação e a correspondência esperada. Se fôr menor que um *threshold* definido é *inlier*, se não é *outlier*.

3. Contar o número de *inliers* para cada iteração com a correspondente transformação rígida (matrizes R e T).

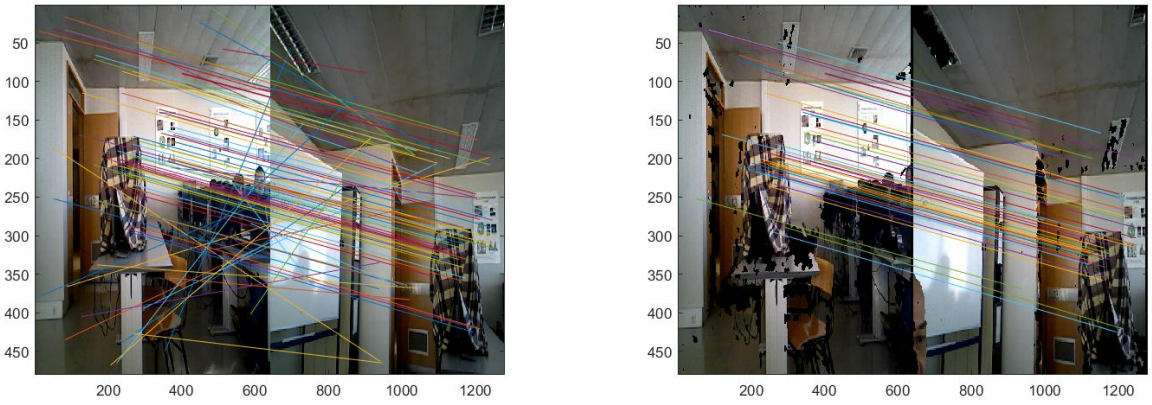
4. Iterar até ao número máximo de iterações. O número de iterações é escolhido em função da seguinte equação:

$$k = \frac{log(1-p)}{log(1-w^n)} \tag{15}$$

em que w^n representa a probabilidade de todos os pontos escolhidos serem *inliers* e p representa a probabilidade pretendida do resultado do *RANSAC* estar correto. Com 500 iterações, assumindo $w^n = 0.4^4$, obtém-se uma probabilidade de o *RANSAC* estar correto de aproximadamente 100% (é impossível garantir um resultado certo dado que $p < 1$).

5. No final de percorrer todas as iterações, resolver um problema de otimização de R e T com todos os dados que foram considerados *inliers*, correspondentes à iteração que obteve o maior número de *inliers*.

Na figura 8, apresentam-se todas as correspondências entre os pontos chave, que foram considerados *inliers*, das duas imagens.



(a) Feature matching usando SIFT (1.5 threshold)

(b) Matching final usando RANSAC (0.3 threshold)

Figura 8: A escolha dos pontos correspondentes é feita à custa do método *RANSAC*, que escolhe o modelo que maximiza o número de inliers.

No final, com o conjunto dos *inliers*, resolve-se o problema de otimização enunciado na equação 13, obtendo-se as matrizes R e T que minimizam o erro quadrático, tal como pretendido. Para testar as matrizes R e T obtidas, aplicou-se a transformação aos pontos pertencentes à *pointcloud* obtida pela câmara 2, combinando as *pointclouds* das duas câmaras obtendo-se o resultado identificado na Figura 9.

3.2.2 Correspondência entre objetos detetados pelas duas câmaras

Através dos métodos relativos à parte 1, anteriormente explicados, tem-se para cada câmara as coordenadas dos vértices de cada caixa, bem como a sequência das imagens que indicam a presença de determinado objeto. Com os parâmetros da transformação rígida calculados, aplica-se a transformação aos vértices das caixas obtidas para a câmara 2, ficando com as coordenadas dos vértices das caixas que envolvem os objetos detetados pela câmara 2, representadas no sistema de coordenadas da câmara 1 (mundo). De seguida, percorrem-se os objetos obtidos com a câmara 1 e para cada um, verifica-se se existem objetos da câmara 2 em que pelo menos

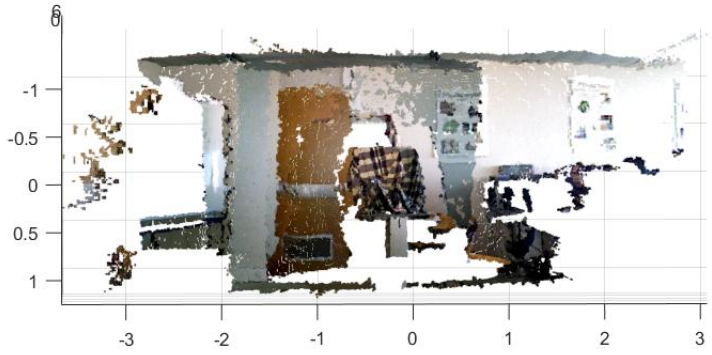


Figura 9: Reconstrução final usando Procrustes Analysis a todos os inliers

uma das *frames* em que aparecem coincidem. Calcula-se a distância entre os centros de massa das caixas entre o objeto da câmara 1 e todos os objetos da câmara 2, no sistema de coordenadas do mundo, que tenham pelo menos uma *frame* coincidente. Se a interseção da deteção entre objetos resultar em vários *frames*, realiza-se uma média de todas as distâncias. Após calcular todas as distâncias, se a distância mínima for menor que um *threshold*, faz-se correspondência entre os 2 objetos. No final, todos os objetos detetados pelas câmaras 1 e 2 que não tenham correspondência são considerados objetos que apenas 1 das câmaras detetou.

4 Limitações dos métodos utilizados

Os métodos utilizados para cumprir os objetivos propostos eram, na grande maioria, simples. Têm a vantagem de ser fáceis de implementar, no entanto, não cobrem todos os casos possíveis ocorrendo más classificações em casos de oclusão, por exemplo, pelo que numa aplicação pouco tolerante a erros seria necessário utilizar métodos mais complexos. Outro exemplo de um erro obtido devido ao uso de um método simples para obter o *background* encontra-se na figura 10. O *background* é calculado através da mediana de todo o conjunto de imagens. Neste *dataset* é frequente aparecerem pessoas na posição assinalada na figura, pelo que é considerado no *background* a forma de uma pessoa. Com o uso de outros métodos mais complexos, este erro não aconteceria. Neste *dataset* em particular adicionou-se artificialmente imagens sem objetos (efetivamente *background*) para resolver este problema.

Outro problema dos métodos utilizados foi a necessidade de definir e afinar parâmetros de *threshold*. Estes parâmetros foram afinados com base nos conjuntos de dados de treino, no entanto, num conjunto de dados de teste independente é provável que os parâmetros que otimizem o bom funcionamento da aplicação sejam diferentes dos obtidos com o conjunto de dados de treino. Por exemplo, é necessário um *threshold* correspondente à distância máxima (de cor e métrica) de um objeto aos restantes, para verificar se esse objeto é ruído. No caso de o ambiente ser uma sala de aula, é natural que esse *threshold* tome um valor bastante menor, quando comparado com um ambiente de uma pista de fórmula 1, dado que é expectável que a distância entre um objeto presente em duas imagens consecutivas seja muito maior na segunda

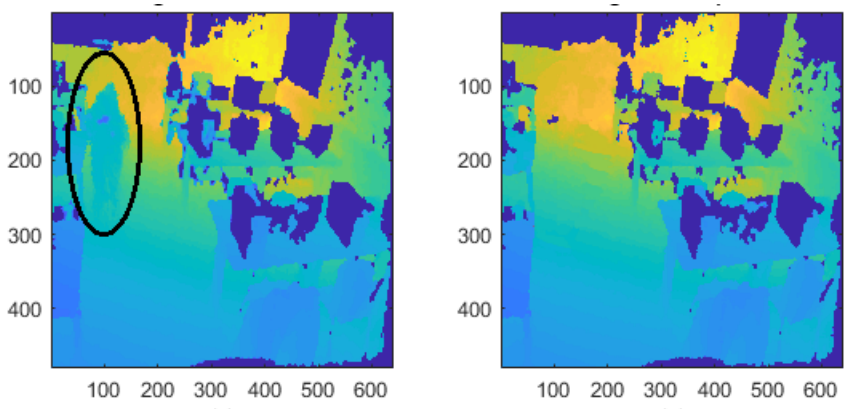


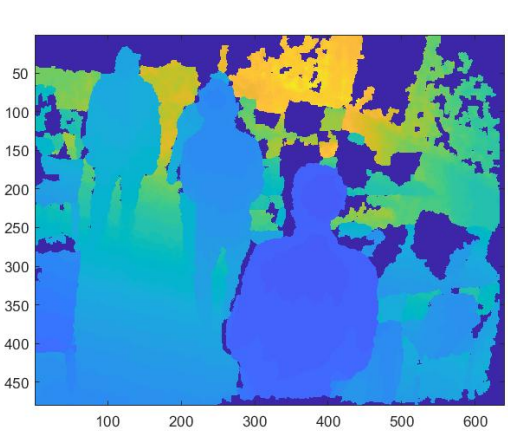
Figura 10: Representação do *background* obtido com a mediana (esquerda) e do esperado (direita)

situação. No entanto, ao saber a finalidade da aplicação conseguem-se afinar estes parâmetros, de forma a que apresente o comportamento esperado para essa situação. Além disto, para facilitar o comportamento dos métodos utilizados, assumiram-se hipóteses. Por exemplo, como já foi referido, assumiu-se que os objetos a detetar não tomam dimensões muito pequenas (menores que 700 píxeis), de forma a ser eliminada uma grande parte do ruído e a evitar o seu processamento.

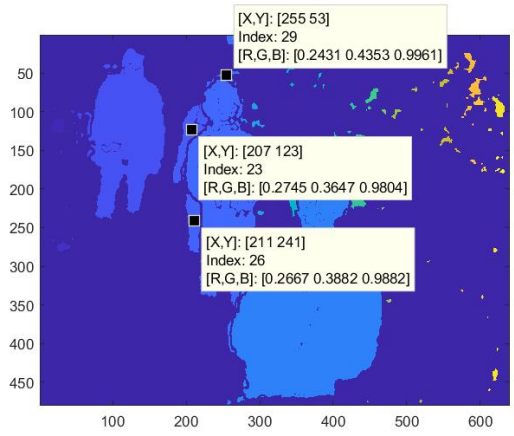
5 Análise experimental nos *datasets* de treino fornecidos

Na figura 11 apresenta-se um erro de identificação de objetos devido a oclusão. Como se pode verificar, uma das pessoas está à frente de outra, sendo que a câmara não consegue detetá-la totalmente, mas apenas um braço, uma perna e uma parte da cabeça. Como resultado, o programa considera numa fase inicial, estas partes da pessoa como 3 possíveis objetos diferentes. No entanto, na *frame* anterior, não haverá nenhum objeto com semelhantes caraterísticas de cor e de centro de massa, pelo que estes 3 possíveis objetos poderão ser descartados, ou somente um deles ser escolhido, modificando o resultado final.

Na figura 12 (a), verifica-se a presença de duas pessoas que estão aproximadamente com o mesmo valor da coordenada z (profundidade) e próximas uma da outra. Devido a isso, na imagem de identificação dos objetos, essas duas pessoas são consideradas a mesma, o que não deveria acontecer. Este erro deve-se a ruído que aparece entre os dois objetos e poderia ser retirado aumentando o tamanho do disco usado na filtragem morfológica. No entanto, este aumento poderia separar, noutros casos, um objeto em duas partes. Por exemplo, poderia separar a cabeça de uma pessoa do resto do corpo. No entanto, noutros casos em que duas pessoas estão lado a lado em toda a extensão do corpo, esta alteração não suritira efeito, e seria necessário utilizar outros métodos de segmentação, como por exemplo gradiente de intensidade, que também podem não ser eficazes em casos de objetos da mesma cor.

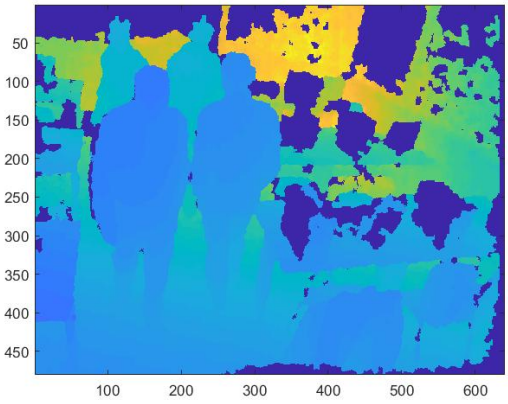


(a) Imagem *depth* com uma pessoa atrás de outra

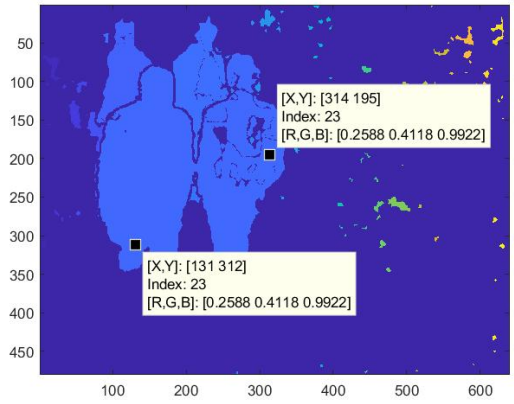


(b) Imagem de identificação dos objetos

Figura 11: Limitação do programa onde se verifica um erro de identificação de objetos devido a oclusão



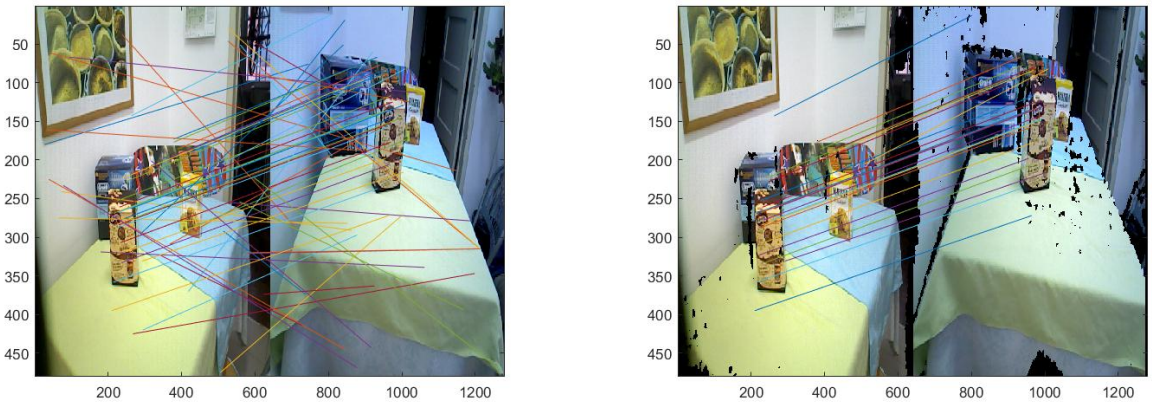
(a) Imagem *depth* com duas pessoas lado a lado



(b) Imagem de identificação dos objetos

Figura 12: Limitação do programa onde se verifica um erro de identificação de objetos devido à junção de 2 pessoas diferentes num só objeto

No exemplo da Figura 13, dadas as dimensões reduzidas da cena, é necessário ajustar o *threshold* associado ao erro do *RANSAC*, de forma a classificar *outliers* corretamente. Utilizando o *threshold* padrão de 0.3, utilizado nos *datasets* de *tracking* na sala do laboratório, o *RANSAC* considerava como inliers más correspondências, nomeadamente correspondências entre a parede e a mesa, cuja distância é inferior a um *threshold* de valor elevado.



(a) Feature matching usando SIFT (1.5 threshold)

(b) Matching final usando *RANSAC* (0.15 threshold)

Figura 13: Neste exemplo foi necessário ajustar o *threshold* do RANSAC à dimensão do problema
Com um valor correto do *threshold*, a reconstrução final apresenta bons resultados, com erros inferiores a 5cm, como se pode verificar na Figura 14

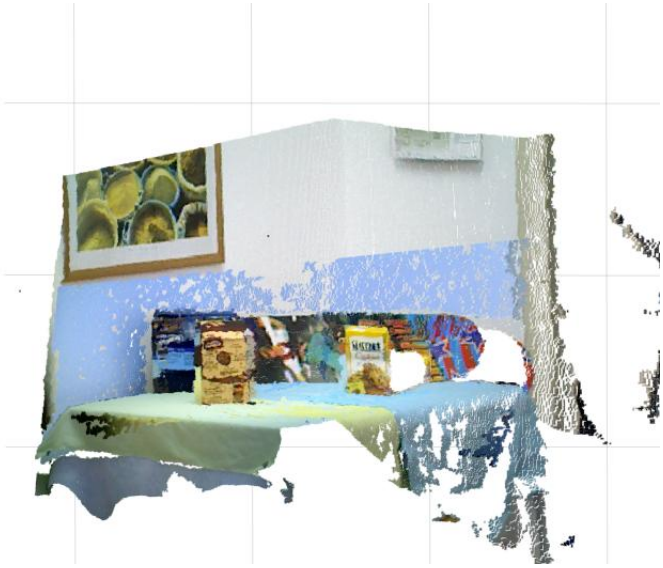
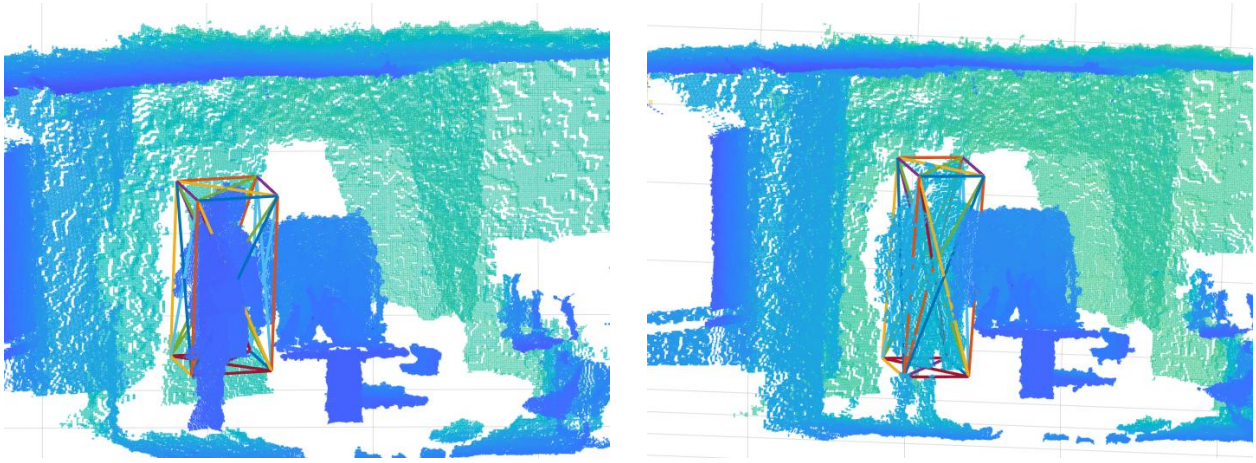


Figura 14: Reconstrução final usando Procrustes Analysis a todos os inliers

Na figura 15, encontram-se duas *pointclouds* obtidas após a reconstrução das duas câmaras, em duas *frames* consecutivas.

Como se pode verificar, o programa é capaz de detetar e encapsular razoavelmente o objeto detetado. Verfica-se em ambos os casos que os membros inferiores não estão encapsulados dentro da caixa, o que pode ser explicado pelo facto de este ser filtrado dado que as diferenças de profundidade são muito reduzidas perto do chão. Por outro lado, na figura 15 (b), verifica-se que alguns pontos à esquerda associados ao objeto não estão encapsulados pela caixa. Este erro deve-se essencialmente ao erro de reconstrução, que transformou os vértices da caixa da câmara 2 para a câmara 1. Este erro pode ser verificado observando em detalhe o objeto em questão, em que se verifica um erro de reconstrução aproximado de 10 cm no objeto.



(a) Um dos objetos detetados

(b) O mesmo objeto detetado noutro *frame*

Figura 15: Deteção de um objeto no *dataset* lab1, com duas câmaras estacionárias

(a) O primeiro frame onde o objeto é detetado na realidade

(b) Frame seguinte, onde o objeto é detetado e seguido pelo programa

Figura 16: Limitação do programa onde se verifica atraso na primeira deteção do objeto

No exemplo demonstrado na figura 16 relativo ao mesmo *dataset*, apresenta-se duas *point-clouds* obtidas com a mesma câmara em 2 *frames* consecutivas. Verifica-se que apesar de ser a mesma pessoa, o programa considera dois objetos diferentes. Esta diferença é explicada pelo valor(custo) de semelhança ser superior ao *threshold* definido, resultado de um valor de diferença de cor elevado pelo que a cabeça é detetada mas não é associada a nenhum objeto. Portanto, o objeto em questão só é detetado pelo programa na *frame* seguinte, onde já se detetam características como a cor da roupa. Esta limitação verifica-se em todos os casos estudados onde esta situação ocorreu. Uma alteração dos parâmetros relativos à correspondência entre objetos em cada frame poderia resolver este problema, mas possibilitava a introdução de correspondências erradas noutros problemas.

Referências

[1] A. Bhattacharyya, "On a measure of divergence between two statistical populations defined by their probability distributions," *Bulletin of the Calcutta Mathematical Society*, vol. 35, p. 99–109, 1943.

[2] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, 2004.

[3] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, pp. 381–395, 06 1981.