

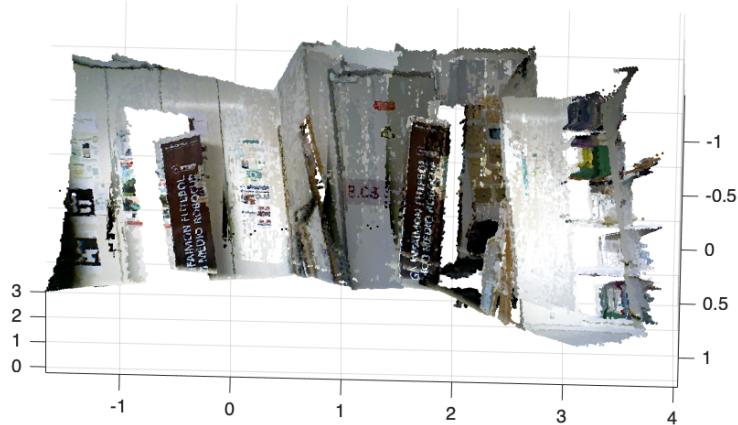


Instituto Superior Técnico

Processamento de Imagem e Visão

Engenharia Electrotécnica e de Computadores

Fusão de nuvens de pontos e detecção de objectos em movimento



Grupo 26

Autores

David Souto
Marcelo Jacinto
Carolina Santos

Número de estudante

86966
87063
87716

Ano lectivo 2019/2020 - 1º semestre

1 Introdução

Sensores como o kinnect ou intel real sense contêm um conjunto de camaras RGB e de profundidade, como se pode verificar na figura 1[a]. Estas câmaras capturam imagens com a mesma resolução, como se pode verificar na figura 1[b]. Este tipo de sensores têm crescido em popularidade ao longo dos anos, pois combinando informações deste par de imagens é possível reconstruir uma cena em 3D mapeando as cores RGB aos respectivos pontos 3D obtidos a partir da imagem de profundidade.

O objetivo deste projeto é reconstruir uma cena em 3D usando múltiplos pares de imagens adquiridas com sensores semelhantes ao kinnect. Em adição à reconstrução de uma cena 3D, pretende-se também identificar objectos que se tenham deslocado entre capturas sucessivas de imagens usadas para a reconstrução da cena.

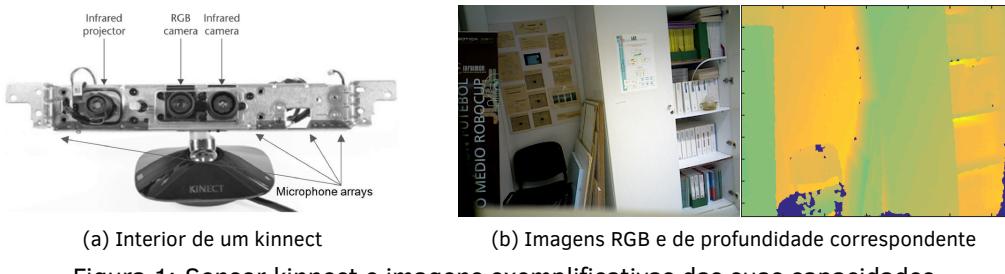


Figura 1: Sensor kinnect e imagens exemplificativas das suas capacidades

2 Formulação do Problema

Para cada conjunto de imagens RGB e profundidade, é necessário calcular a posição e orientação do kinnect em relação às coordenadas da cena. Depois de efetuado este passo, pode-se proceder à fusão da nuvem de pontos, tendo para isso de calcular a matriz de rotação e translação entre as nuvens de pontos. Após este passo, já com a matriz de rotação e translação entre nuvens de pontos pode-se efetuar a detecção de objectos em movimento entre as sucessivas capturas.

De forma a efetuar a **fusão de nuvens de pontos**, dada uma sequência de imagens RGB e de profundidade, têm de efetuar os seguintes passos:

1. Para cada conjunto de imagens RGB e profundidade:
 - (a) Obter, a partir da imagem de profundidade, para cada coordenada $[x, y]^T$ na imagem, a correspondente coordenada $[X, Y, Z]^T$ no referencial do mundo;
 - (b) Obter, a partir da imagem RGB, os valores RGB correspondentes a cada ponto $[X, Y, Z]^T$ no referencial do mundo;
 - (c) Com os vetores de pontos $[X, Y, Z]^T$ e $[RGB]^T$ correspondentes é possível construir uma nuvem de pontos.
2. Seja $j \in \{1, \dots, N\}$, sendo N o numero de nuvens de pontos a fundir. Encontrar correspondência entre pontos $[X, Y, Z]^T$ da nuvem j e pontos $[X, Y, Z]^T$ da nuvem $j+1$. Como encontrar correspondências entre pontos em 3D é complicado, tira-se então partido das imagens RGB j e $j+1$ que deram origem às respectivas nuvens de pontos. Com essas imagens RGB usa-se um algoritmo de "feature detection" e de "matching" para obter pontos $[x, y]^T$ na imagem j que correspondem a pontos $[x, y]^T$ na imagem $j+1$. De seguida, obtém-se os pontos $[X, Y, Z]^T$ em 3D correspondentes aos pontos 2D, $[x, y]^T$, em cada uma das imagens;
3. Calcular as transformações entre nuvens de pontos sucessivas (rotações e translações), isto é, da nuvem de pontos $j+1$ para a nuvem de pontos j tirando partido dos matches entre pontos $[X, Y, Z]^T$.

4. Dadas as transformações entre nuvens de pontos sucessivas, isto é, adquiridas a partir de imagens adquiridas sucessivamente, propagar as transformações de forma a representar todas as nuvens de pontos no mesmo referencial;
5. Aplicar um algoritmo iterativo de modo a reduzir o erro entre transformações;

Para efectuar a **detecção de objectos em movimento** entre cenas capturadas sucessivamente, podem-se seguir os seguintes passos:

1. Para cada conjunto de imagens de profundidade j e $j+1$:
 - (a) A partir da imagem de profundidade $j+1$, para cada coordenada $[x, y]^T$ na imagem, obter a correspondente coordenada $[X, Y, Z]^T$ no referencial da nuvem de pontos $j+1$;
2. Aplicar a matriz de rotação e de translação, calculadas anteriormente, por forma a obter os pontos $[X, Y, Z]^T$ representados no referencial da nuvem de pontos j . A partir dos pontos $[X, Y, Z]^T$ no referencial j , obter a imagem de profundidade $j+1$ no referencial da câmara j .
3. Verificar quais dos pontos são comuns à duas imagens de profundidade.
4. Para os pontos comuns analisar a diferença do valor de profundidade entre os pontos da câmara 2 mapeados na câmara 1 e dos pontos realmente detectados pela câmara 1.
5. Uma diferença significativa nos níveis de profundidade indica que se está perante um objecto móvel.
6. Para os pontos $[x, y]^T$ onde foi detetado um objecto móvel, obter os correspondentes pontos $[X, Y, Z]^T$ e aplicar uma rotação e translação para que estes sejam representados no referencial da nuvem de pontos 1.

3 Nuvem de pontos a partir de Imagens RGB e de profundidade

3.1 Modelo da Camara

Cada camara, RGB ou de profundidade projecta pontos $\mathbf{X} = [X, Y, Z]^T$ no referencial do mundo para o plano da imagem $\mathbf{x} = [x, y]^T$, tal como representado na figura 2.

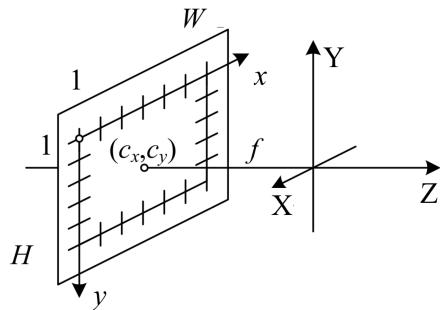


Figura 2: Modelo da câmara, adaptado de [2]

Considere-se o modelo da câmara completo, que pode ser representado em coordenadas homogéneas por:

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} fs_x & 0 & c_x \\ 0 & fs_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (1)$$

sendo os parâmetros intrínsecos da camara dados por: f a distância focal, s_x e s_y factores de escala em x e em y respetivamente (em pixel/m), c_x e c_y as coordenadas do ponto principal (em pixels). Os parâmetros extrínsecos são constituídos pelas matrizes de rotação, R , e de translação, T , do referencial do mundo para o referencial da camara.

Fica assim estabelecido um método que permite a partir de pontos numa cena 3D obter os correspondentes pontos 2D numa imagem e vice-versa.

3.2 Obtenção de pontos no referencial do mundo

Para se obter os pontos $[XYZ]^T$ no referencial do mundo tira-se partido do modelo da câmara definido previamente, e arbitra-se que o referencial do mundo coincide com o referencial da câmara de profundidade. Desde modo, a matriz de rotação corresponde à matriz identidade ($R = I$) e a matriz de translação a zero ($T = [0, 0, 0]$). Desta forma podemos manipular a equação 1 de forma a obter a equação 2, sendo que o valor de Z é retirado da imagem de profundidade.

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = [K]^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (2)$$

No final de termos os valores de $[X, Y, Z]^T$ obtém-se o vetor $[x, y]^T$ correspondente na imagem RGB usando para isso a equação 1, de forma a obterem-se os valores RGB respectivos a cada ponto $[X, Y, Z]^T$. Repetindo este processo para todos os pontos na imagem de profundidade, obtém-se um vetor de coordenadas $[X, Y, Z]^T$ e um vetor $[R, G, B]^T$ com as cores correspondentes. Pode-se então verificar na figura 3 uma nuvem de pontos produzida a partir de um conjunto de imagens RGB e de profundidade, usando o método descrito previamente.

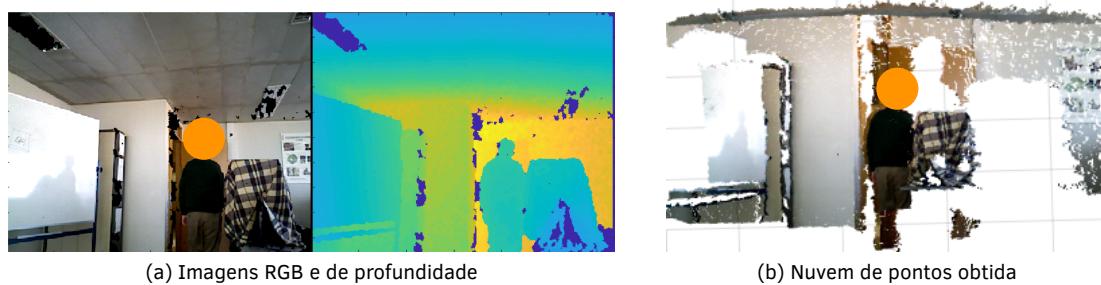


Figura 3: Construção da nuvem de pontos a partir de um par de imagens RGB e de profundidade

4 Fusão de nuvens de pontos

Partindo do princípio que se conhecem os vetores de pontos em 3D e os correspondentes valores RGB para cada uma das N nuvens de pontos que se pretendem fundir, começa-se por determinar os pontos $[X, Y, Z]^T$ na nuvem de pontos $j+1$ que correspondem aos pontos $[X, Y, Z]^T$ na nuvem de pontos j .

Como determinar correspondências entre pontos em 3D é um problema de elevada complexidade, adoptou-se uma aproximação. Esta aproximação consiste em detectar "features" nas imagens RGB que deram origem às nuvens de pontos usando o algoritmo **SIFT**. Usa-se ainda um "matcher" de forma a descobrir quais os pontos $[x, y]^T$ da imagem RGB $j+1$ que correspondem aos pontos $[x, y]^T$ da imagem RGB j . Como os vectores de coordenadas $[X, Y, Z]^T$ são ordenados pelas coordenadas $[x, y]^T$ das imagens RGB, sabe-se quais os pontos $[X, Y, Z]^T$ na nuvem de pontos $j+1$ que correspondem aos pontos $[X, Y, Z]^T$ na nuvem de pontos j .

4.1 Detecção de features e obtenção de correspondências entre imagens

A escolha do algoritmo **SIFT (scale invariant feature transform)** para se efetuar a detecção de "features" nas imagens RGB deve-se ao facto deste ser invariante a mudanças de escala, algo que não acontece em algoritmos como o **Harris corners**.

Tal como descrito em [4], o algoritmo **SIFT** começa por definir um conjunto de escalas e filtra cada imagem com um filtro DoG (difference of Gaussians) aplicado nesse espaço de escalas. Os "keypoints" detectados vão corresponder aos extremos do sinal à saída do filtro. De seguida "keypoints" detectados nos quais o valor do gradiente é pequeno são eliminados, de forma a eliminar "keypoints" que existem em zonas de baixo contraste na imagem.

Após efetuado o processo de detecção de "keypoints", a cada "keypoint" é associada uma orientação que é extraída de um histograma da orientação do gradiente numa janela de 16x16 em torno desse "keypoint".

Por fim é criado um "descriptor". Pegando na janela 16x16 mencionada no passo anterior divide-se esta em janelas 4x4. Para cada janela calcula-se a magnitude do gradiente, a sua orientação e obtém-se histogramas 4x4 de 8 "bins". Com base nos histogramas, gera-se um vetor com 128 features que é usado para caracterizar a "patch"local em torno do "keypoint".

Para efetuar o matching the "keypoints", dados dois vetores de features d_{0i} e d_{0j} extraídos de imagens diferentes podem-se medir as suas semelhanças recorrendo à norma euclidiana: $d_{ij} = \|d_i - d_j\|$. Para cada d_{0i} selecciona-se o vizinho d_{1j} para o qual a distância é pequena e suficiente. Em alternativa podem-se usar dois vizinhos em vez de 1 de forma a obter resultados mais fiáveis.

4.2 Calculo das matrizes de rotação e translação entre nuvens de pontos

Para calcular as matrizes de rotação e translação, optou-se por usar um método que cálcula as matrizes de rotação e translação ótimas para uma transformação rígida recorrendo à decomposição em valores singulares de uma matriz de covariâncias. Este método assume a priori que existem um conjunto de pontos que são correspondentes. Tendo um conjunto de pontos correspondentes p_i e p'_i pretende-se que:

$$p'_i = Rp_i + T \quad (3)$$

Desta forma pretende-se minimizar o erro:

$$\sum_{i=1}^N \|(Rp_i + T) - p'_i\|^2 \quad (4)$$

De forma a resolver o problema de minimização, os seguintes passos devem ser efectuados:

- Calcular os centroides dos dois conjuntos de pontos

$$\bar{p} = \frac{\sum_{i=1}^N w_i p_i}{\sum_{i=1}^N w_i} \text{ e } \bar{p}'_i = \frac{\sum_{i=1}^N w_i p'_i}{\sum_{i=1}^N w_i} \quad (5)$$

sendo w_i o peso atribuído a cada ponto. Neste problema assume-se que todos os pontos têm o mesmo peso.

- Calcular o valor centrado de cada ponto nos 2 conjuntos de pontos:

$$x_i := p_i - \bar{p} \text{ e } y_i := p'_i - \bar{p}' \quad (6)$$

- Obter a matriz de covariâncias e efetuar decomposição desta em valores singulares:

$$S = U\Sigma V^T, \text{ sendo } S = (p_i - \bar{p})(p'_i - \bar{p}')^T \quad (7)$$

De onde no final se retira que:

$$R = VU^T \quad (8)$$

Sendo que se tem de ter particular atenção ao sinal do determinante de R obtido, tal como referido no **anexo B**. Tendo a matriz de rotação é possível obter o vetor de translação.

A dedução deste algoritmo encontra-se no **anexo B** e foi adaptada de [3].

4.3 Implementação do RANSAC

Como a detecção de "features" é feita recorrendo às imagens RGB usando o algoritmo SIFT e o "matching" de "features/descriptors" é feito recorrendo à semelhança entre "features" adquiridas em imagens diferentes, irão existir "outliers" nas várias correspondências identificadas. Para além disto, vale apena salientar que este método para encontrar pontos correspondentes é apenas uma simplificação do problema original - encontrar pontos correspondentes

entre nuvens de pontos. Ao se identificar pontos correspondente em imagens RGB e posteriormente obter os pontos 3D correspondentes pode dar também origem a erros de "matching" que também têm de ser eliminados.

A remoção de "outliers" deverá ser efetuada de forma consistente, pelo que o algoritmo RANSAC (RANdom SAmple Consensus) é aplicado. Este algoritmo consiste em:

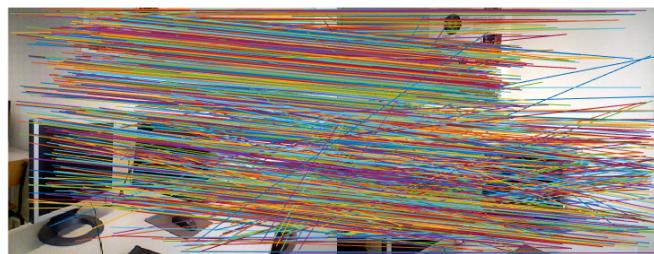
Para k iterações:

- Selecionar 4 "matches" da lista de "matches" de duas nuvens de pontos $j+1$ e j ;
- Calcular a matriz de rotação e de translação usando esses 4 pontos $[X, Y, Z]^T$ correspondentes;
- Calcular para cada ponto p_j na lista de matches da nuvem de pontos j qual o valor p'_j esperado, obtido pela multiplicação de p_j pela matriz de rotação e somado pela respectiva translação;
- Calcular a distância de cada um dos pontos p'_j ao verdadeiro valor p_j ;
- Calcular o número de pontos que respeitam $\|p'_j - p_j\| < \epsilon$, e guardar esses números num vetor de "inliers".

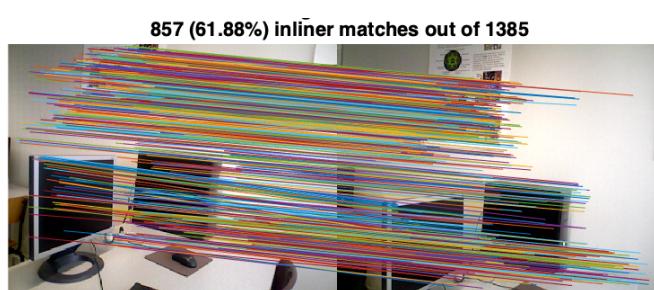
Escolher a transformação com o maior número de inliers;

Calcular as novas matrizes de rotação e translação tendo em conta agora todos os inliers;

Na figura 4 é possível verificar a aplicação dos conceitos introduzidos ao longo da secção 4. Na imagem a), verifica-se que são encontradas um total de 1385 SIFT features comuns a ambas as imagens RGB. Pode-se ainda verificar que algumas das correspondências encontradas pelos algoritmos são outliers, pelo que deverão ser removidos aplicando RANSAC. Na imagem b), verificam-se que apenas 857 "matches" são classificadas como inliers após a aplicação de RANSAC. **NOTA:** a correspondência entre features, após a aplicação do RANSAC encontra-se representada nas imagens RGB e não nas nuvens de pontos, de forma a visualizar de forma mais intuitiva o funcionamento do algoritmo.



(a) Correspondência entre SIFT features pré-RANSAC



(b) Correspondência entre SIFT features pós-RANSAC

Figura 4: Exemplificação do funcionamento do algoritmo RANSAC

4.4 Exprimir todas as nuvens de pontos no mesmo referencial

Devido ao facto de neste projeto, apenas se garantir o "overlap" entre imagens consecutivas, optou-se por calcular as transformações R e T que levam da nuvem de pontos $j+1$ para a nuvem de pontos imediatamente adjacente j e não de todas as nuvens de pontos para todas

as nuvens de pontos. Pretende-se com isto prevenir o caso em que existem "matches" entre imagens que não se sobrepõem devido ao facto destas conterem objectos iguais.

Neste momento assume-se que se têm calculados, N pares de imagens RGB e profundidade, se têm N nuvens de pontos (obtidas a partir desse conjunto N de imagens) bem como as respectivas transformações de referenciais da nuvem de pontos $j+1$ para j . Então:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}_j = \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_1 \\ R_{21} & R_{22} & R_{23} & T_2 \\ R_{31} & R_{32} & R_{33} & T_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}_{j+1} \Rightarrow \tilde{m}_j \sim H_{j+1,j} \tilde{m}_{j+1} \quad (9)$$

Da mesma forma que se tem:

$$\tilde{m}_{j-1} \sim H_{j,j-1} \tilde{m}_j \quad (10)$$

Substituindo uma equação na outra, pode-se verificar que:

$$\tilde{m}_{j-1} \sim H_{j,j-1} H_{j+1,j} \tilde{m}_{j+1} \quad (11)$$

Quer isto dizer que por multiplicações encadeadas de matrizes de transformação, pode-se exprimir todos os pontos das N nuvem de pontos no mesmo referencial. Este é o raciocínio usado para obter R e T que permitem levar pontos da nuvem de pontos expressos num referencial j para o referencial da nuvem de pontos 1.

Na figura 5 encontra-se um exemplo da junção de várias nuvens de pontos, exprimindo todos os pontos no referencial da nuvem de pontos 1.

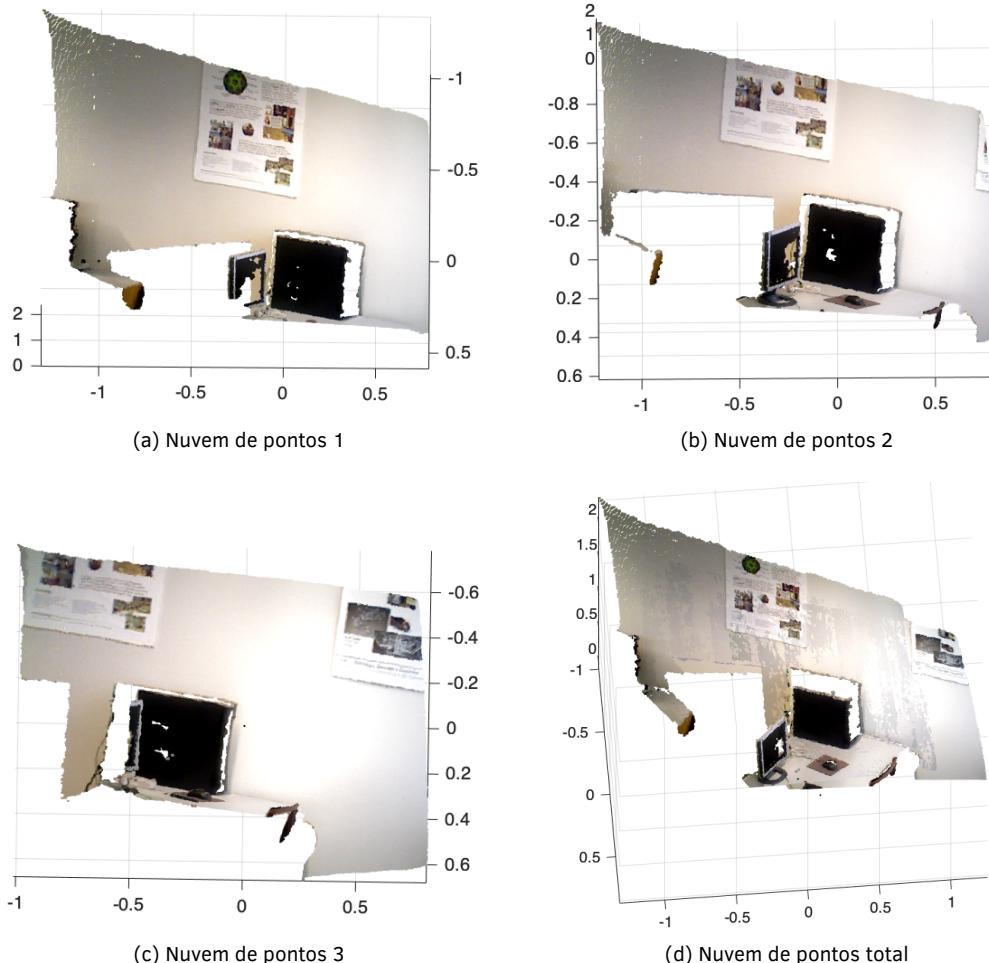


Figura 5: Reconstrução de uma cena 3D a partir de múltiplas nuvens de pontos expressas no mesmo referencial

5 Detecção de objectos em movimento

Nesta secção pretende-se demonstrar o procedimento adoptado para detectar objectos em movimento entre nuvens de pontos sucessivas, isto é, entre nuvens de pontos obtidas a partir de imagens adjacentes. Para isso tem-se por base a matriz de rotação e de translação estimadas na primeira parte deste projecto.

Como resultado desta fase do projecto, apresentam-se todos os pontos móveis $[X, Y, Z]^T$ detectados, pontos estes que juntos constituem os objectos que se moveram.

Este método parte do pressuposto que a matriz de rotação e de translação estimadas caracterizam bem a relação entre referenciais de nuvens de pontos consecutivas, ou seja que o erro na mudança de coordenadas é muito reduzido, o que permite desprezar este mesmo erro ao longo dos cálculos.

5.1 Passos a seguir para a detecção de objectos

De forma a simplificar o problema, considerem-se apenas 2 conjuntos de imagens RGB, representadas na figura 6 e as suas respectivas imagens de profundidade (que não se encontram representadas).



(a) Imagem RGB 1



(b) Imagem RGB 2

Figura 6: Duas imagens RGB na qual um objeto desapareceu e a orientação da camara é diferente

Começa-se por calcular para cada coordenada $[x, y]^T$ na imagem de profundidade 2, a correspondente coordenada $[X, Y, Z]^T$ no referencial do mundo utilizando a equação 2.

Tendo agora todos os pontos $[X, Y, Z]^T$ tem de se efetuar uma mudança de coordenadas da nuvem de pontos 2 para a nuvem de pontos 1 usando a equação 9.

Como é mais fácil detectar diferenças entre pontos em 2D do que em 3D, ou seja, é mais fácil detectar diferenças entre imagens do que em nuvens de pontos, é efetuada mais uma passagem, de modo a poder exprimir-se a imagem de profundidade 2 no referencial da camara 1, usando os pontos $[X, Y, Z]^T$ expressos no referencial da nuvem de pontos 1 e aplicando a estes a equação 2 novamente.

Nesta etapa todos os pontos da imagem de profundidade 2 são expressos no referencial da imagem de profundidade 1. Contudo, nem todos os pontos da imagem 2 são visto pela imagem 1, visto que existe a possibilidade de as imagens terem sido tiradas de perspectivas diferentes. Significa então que irão existir pontos da imagem 2 que não terão correspondência dentro dos limites visíveis da imagem 1 e vice-versa. Havendo esta incoerência, é preciso identificar apenas os pontos que são comuns entre as duas imagens de profundidade, removendo tanto os pontos da imagem 2 fora do alcance da imagem 1 assim como os pontos da imagem 1 que não são expressos na imagem 2.

A remoção dos pontos não comuns, torna-se fácil, pois sabendo que todas as imagens têm um tamanho fixo de 640x480 pixels, todos aqueles pontos que mapearem fora destes valores, serão pontos não comuns entre as duas imagens. Na figura 7 a) pode-se verificar a imagem de

profundidade 1 apenas com os pontos comuns à imagem 2 (expressa no referencial da câmara 1). Na figura 7 b) pode-se verificar a imagem de profundidade 2 apenas com os pontos comuns à imagem 1 (expressa no referencial da câmara 2).

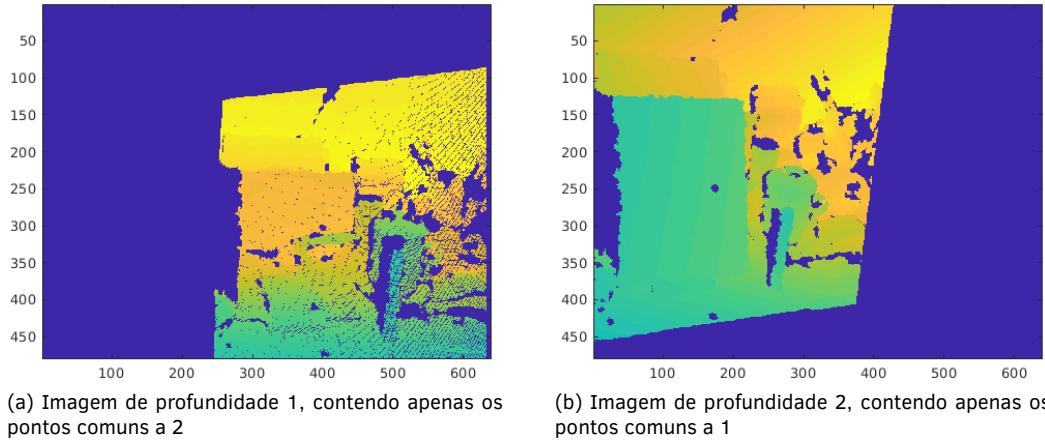


Figura 7: Imagens de profundidade 1 e 2 apenas com os pixels comuns a cada uma das imagens

Removidos todos os pontos que estão fora dos limites de cada imagem, resta agora perceber que objectos se moveram de uma imagem para a outra. Para isso usa-se a informação da coordenada Z obtida através da câmara de profundidade. Subtrai-se então os valores de profundidade, Z , entre os pixels da imagem de profundidade 2 e os correspondentes pixels da imagem de profundidade 1. Idealmente, se um objeto não se tivesse movido, o resultado desta subtração seria 0. No entanto, visto que existe ruído na estimativa das matrizes de rotação e translação usadas anteriormente, bem como na calibração das camaras, este resultado é irrealista. Considera-se portanto que um objeto se moveu se a subtração de profundidades indexadas por pontos correspondentes nas duas imagens for superior em módulo a 0.1.

Para se identificar se o objeto que se moveu se encontrava na imagem 1 ou na imagem 2 é necessário, para o pixel onde foi detetado movimento verificar em qual das duas imagens o valor de Z é menor. O que tiver menor valor de Z corresponde ao objeto, que se moveu, pois está mais perto da câmara. Repetindo esta identificação para todos os pixels comuns às duas imagens, podem-se criar duas imagens virtuais usando valores de 1s e 0s em vez de Z , colocando-se 1 no pixels onde foi detectado um objeto que se moveu na imagem correspondente e 0 caso contrário. Nas figuras 8 a) e b) as imagens virtuais que representam os pixels onde foi detectado movimento em cada uma das imagens.

Como existe ruído na detecção dos objectos, uma das estratégias usadas para reduzir o mesmo é aplicação filtros mediana nas duas imagens virtuais criadas usando uma janela 3x3. Desta forma reduz-se o ruído do tipo "Salt and Pepper".

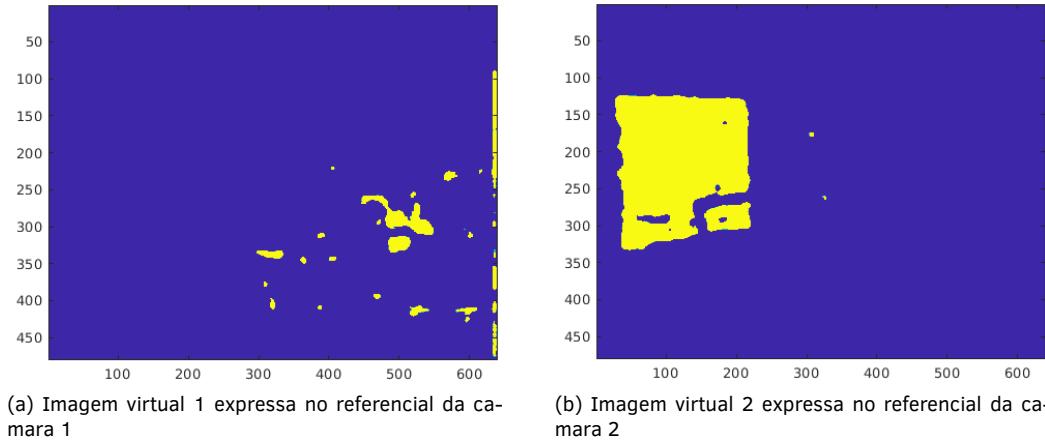


Figura 8: Imagens virtuais que indicam os pixels nos quais foi detectado movimento

De seguida resolve-se um problema de conectividade. Para as duas imagens virtuais, são identificados quais os pontos que são conexos e identificam-se cada conjunto de pontos conexos. Quando um grande conjunto de pontos é conexo significa que se está na presença de uma objeto e não apenas de ruído que não foi eliminado pelo filtro mediana. Neste caso é razoável definir que quando existem pelo menos 15 000 pontos conexos, estamos na presença de um objeto.

Por cada objeto identificado em cada uma das imagens virtuais, obtém-se as respectivas coordenadas $[X, Y, Z]^T$ nas nuvens de pontos, podendo assim reconstruir em 3D os objectos que se moveram e também identificar em que frames ocorreu o movimento. Na figura 9 pode-se verificar que apenas o livro é detetado como um objeto. Pode ainda verificar-se que parte do livro é "cortada", e tal deve-se ao facto de o canto inferior direito do livro não ser identificado como conexo aos restantes pontos do livro, e, não tendo uma densidade de pontos que lhe permita ultrapassar o threshold definido, este não é representado.

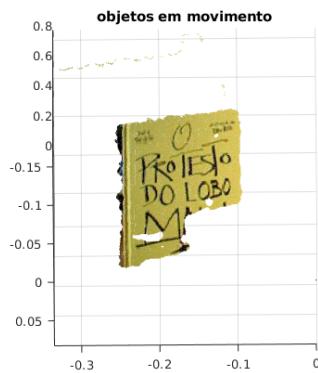


Figura 9: Objecto identificado na imagem 2 expresso no referencial da nuvem de pontos 1

Em termos práticos, a implementação que foi feita usa um sistema de indexes que guarda, quando é feita a passagem de pontos $[X, Y, Z]^T$ correspondentes à nuvem de pontos 2 expressos no referencial na nuvem de pontos 1 para pontos $[x, y]^T$ expressos no referencial da camara 1, a posição (no vetor) correspondente a $[X, Y, Z]^T$ para dado ponto $[x, y]^T$. Desta forma, quando um objeto é identificado na imagem de profundidade, não é necessário aplicar a equação 2 reduzindo o acumular de erros numéricos inerentes das mudanças de referenciais.

Extrapolando o raciocínio descrito para N imagens, efectuando o processo sempre entre imagens adjacentes é possível obter um conjunto de pontos em 3D que representam objectos que se moveram e ainda identificar em que frames esse movimento foi detetado.

6 Valores arbitrados para os algoritmos

Durante a implementação do algoritmo que é descrito ao longo deste relatório algumas decisões tiveram de ser tomadas, nomeadamente no que diz respeito a parâmetros dos algoritmos SIFT e RANSAC.

Durante a implementação dos algoritmos recorreu-se à utilização da biblioteca VLFeat para a detecção das SIFT features e match dos SIFT descriptors. Alguns dos parâmetros pré-definidos não eram os mais adequados ao projeto em questão. Desta forma optou-se por alterar o valor do número de níveis por oitava para 50 (o valor pré-definido era 3). Este aumento, provoca um aumento computacional mas leva à identificação de mais features por imagens, comparativamente ao valor pré-definido.

A obtenção de mais "features" por imagem não é um problema, mesmo que isto leve a que posteriormente ocorram mais outliers nos "matches" inter-imagens. Tal deve-se ao facto de ser implementado o algoritmo RANSAC que irá remover esses "outliers".

Para o RANSAC, em cada iteração usam-se 4 amostras de pontos que são "matches" em ambas as nuvens de pontos (número mínimo necessário para calcular a matriz de rotação e translação que no total têm 12 incógnitas). Para definir se um ponto é considerado "inlier" ou

"outlier" usa-se a distância geométrica entre a posição prevista e a posição real de pontos. Se o ponto previsto tiver uma distância geométrica maior que 0.05m do ponto real, então este será um "outlier". Para o número de iterações optou-se por usar $k = 500$. Assumindo que se tem uma probabilidade de inliers de apenas 50%, usando a fórmula $P = 1 - (1 - p^n)^k$, obtém-se uma probabilidade de sucesso após 500 iterações de $\approx 99.9\%$.

Relativamente ao algoritmo iterativo que visa melhorar as matrizes de transformações entre nuvens de pontos, foi usada uma implementação do ICP (iterative closest point) "out of the box" que o MATLAB fornece. Antes da aplicação deste algoritmo faz-se um "downsample" das nuvens de pontos usando uma grelha de 0.01m. O número máximo de iterações que o algoritmo pode fazer são 5, visto que as matrizes de rotação e translação obtidas previamente são geralmente boas aproximações. Assume-se ainda que apenas 10% dos pontos são inliers, caso contrário o algoritmo poderia começar a divergir, aproximando pontos que na realidade não são correspondentes. O algoritmo ICP é aplicado segundo o processo:

- Corre-se o algoritmo ICP para a transformação da segunda nuvem de pontos para a primeira nuvem de pontos, dando como iniciação a matriz de transformação com os parâmetros de rotação e translação calculados previamente;
- Representa-se a nuvem de pontos 2 no referencial da nuvem de pontos 1, usando a nova matriz de transformação. Acumulam-se os pontos de ambas as nuvens de pontos num vetor;
- Corre-se o algoritmo ICP para a transformação da terceira nuvem de pontos, para a nuvem de pontos acumulada no vetor anterior. Representa-se a nuvem de pontos 3 no referencial da nuvem de pontos 1, usando a nova matriz de transformação. Acumulam-se os novos pontos no vetor de pontos já existem;
- Repetir até não existirem mais nuvens de pontos para processar.

Conclusão

Ao longo deste relatório é descrito um processo para fusão de nuvens de pontos adquiridas através de múltiplos pares de imagens RGB e de profundidade. São também mostrados alguns exemplos de resultados obtidos usando os algoritmos descritos. No entanto, mais resultados são demonstrados no **Anexo A** devidamente comentados.

É importante referir que o método usado para a fusão de nuvens de pontos não é perfeito e é suscetível a problemas que não foram mencionados previamente. Um desses problemas surge quando não são encontradas "matches" suficientes entre imagens RGB sucessivas pelo que se torna impossível, fazer a fusão das nuvens de pontos. Outro problema, surge do facto de se usar um threshold para definir se pontos nos quais foram detectados movimento pertencem a um objeto ou não, pois isto pode levar a que objectos de pequena dimensão não sejam detectados. Também erros na estimação das matrizes de rotação ou translação entre nuvens de pontos podem levar a que grandes planos sejam detectados como objectos que se moveram, ainda que na realidade sejam objectos estáticos.

Referências

- [1] Slides PIV, José Santos Vitor e Jorge-Marques, 2019
- [2] Computer Vision: Algorithms and Applications, Richard Szeliski 2010
- [3] O. Sorkine-Hornung and M. Rabinovich, Least-Squares Rigid Motion Using SVD, (2017)
- [4] <https://towardsdatascience.com/sift-scale-invariant-feature-transform-c7233dc60f37>

Anexo A

Aplicação do algoritmo ao dataset piv2

Como se pode ver pela figura 10 a), a aplicação do algoritmo ao conjunto de imagens **newpiv2** origina resultados bastante positivos, obtendo-se 2 placares individuais, tal como era suposto. Analisando para a figura 10 b) verifica-se que os grandes planos das nuvens de pontos se sobrepõem de forma quase perfeita. Neste dataset, nenhum objeto em movimento foi detetado.

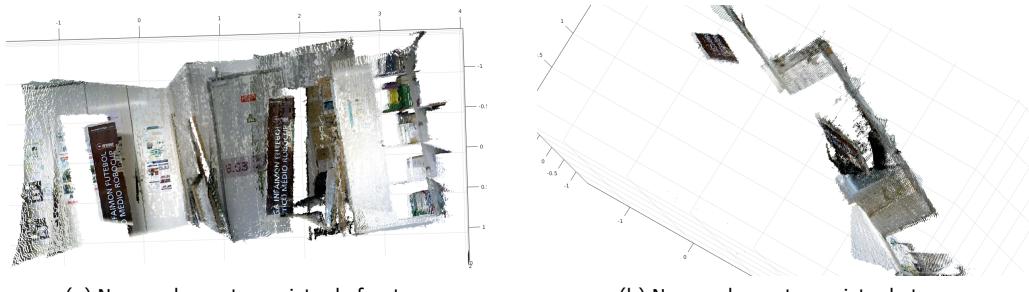


Figura 10: Fusão de nuvens de pontos usando o "dataset"**newpiv2**

Aplicação do algoritmo ao dataset officenmotion

Pode-se ver pelas figuras 11 a) e b) que o algoritmo alinha as nuvens de pontos tirando partido de features encontrados no livro, nas fotos que se encontram nas paredes e nos papeis. Por consequência a nuvem de pontos, vista de frente parece bastante bem alinhada, mas quando vista de topo, pode-se ver que pequenos objectos não são alinhados correctamente, mas que grandes planos se encontram completamente sobrepostos. Na figura 11 c) pode-se verificar os vários objetos detetados entre várias imagens expressos no mesmo referencial. Como esperado, como o livro desaparece e aparece na mesma posição, em vários frames, os objetos detetados, quando expressos no mesmo referencial ficam sobrepostos. Quando vistos em referenciais das nuvens de pontos onde foram detetados, pode-se verificar que foram detetados 2 movimentos do livro.

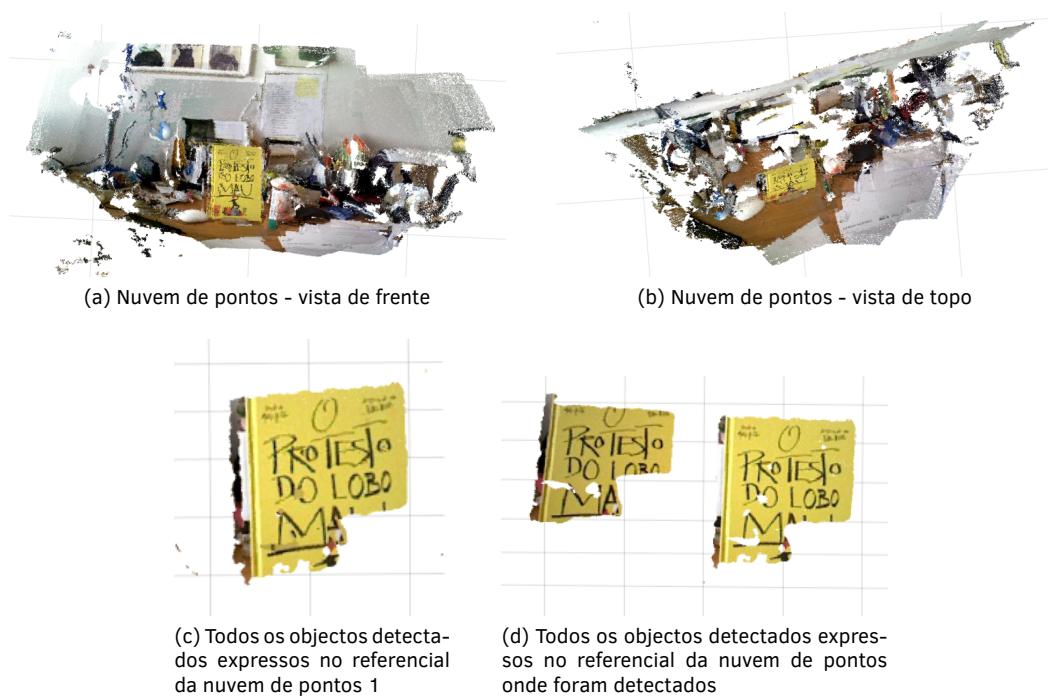


Figura 11: Fusão de nuvens de pontos usando o "dataset"**officenmotion**

Aplicação do algoritmo ao dataset officewmotion

Pela figura 12, pode-se verificar que os resultados obtidos quando há movimento da camara são semelhantes aos resultados obtidos quando não há movimento da camara.



Figura 12: Fusão de nuvens de pontos usando o "dataset" **officewmotion**

Aplicação do algoritmo ao dataset board1

Os resultados obtidos usando o dataset **board1** estão dentro do esperado, pelo que se pode ver pela figura 13 onde os planos representados nas nuvens de pontos se sobrepõem com erro inferior a 5 mm.

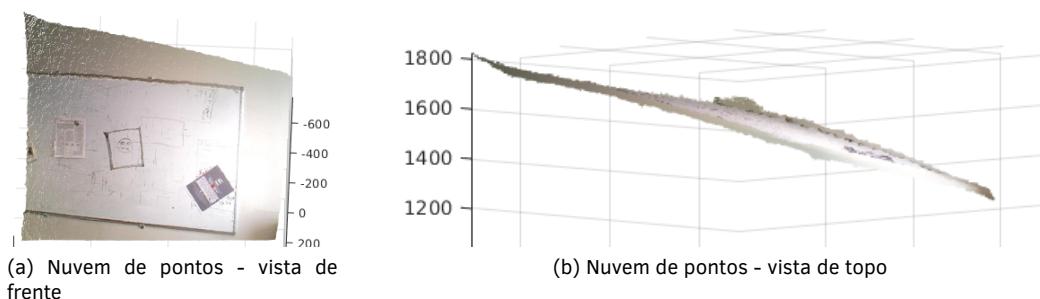


Figura 13: Fusão de nuvens de pontos usando o "dataset" **board1**

Aplicação do algoritmo ao dataset andar_sala

Neste dataset o algoritmo faz as fusões das nuvens de pontos como esperado, como se pode ver pelos grandes planos que se encontram bem alinhados. Pode-se ainda verificar os vários objectos detectados entre imagens à medida que o professor se desloca na sala.

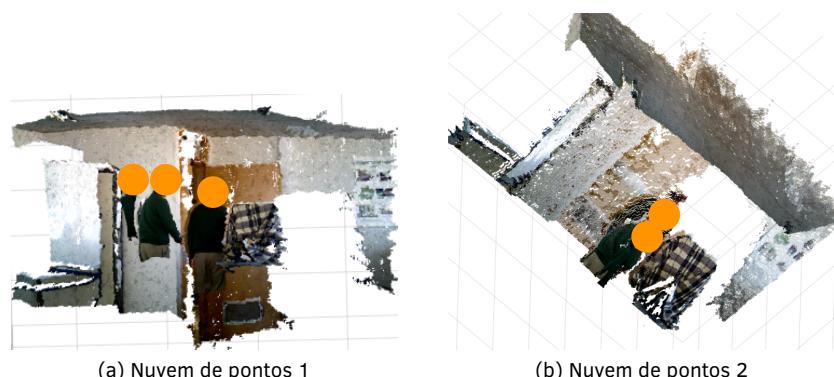


Figura 14

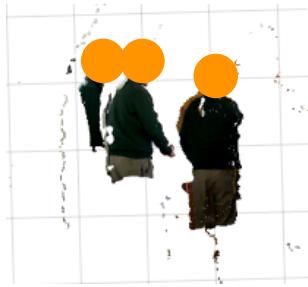


Figura 15

Aplicação do algoritmo ao dataset room1

Neste dataset é visível, entre o plano do tecto e o quadro azul, que existe um ligeiro desalinhamento de duas das nuvens de pontos fundidas. Por consequência uma parte do plano junto ao quadro azul é detetado como um objeto móvel, quando não deveria.

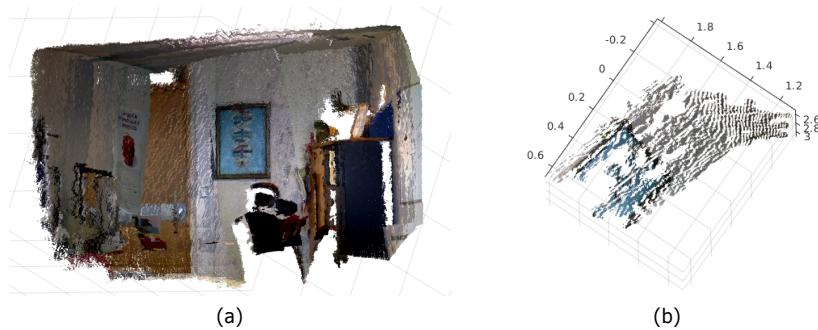


Figura 16: Fusão de nuvens de pontos usando o "dataset" **room1**

Aplicação do algoritmo ao dataset lab_piv_short

Da mesma maneira que no dataset anterior um dos planos não ficou perfeitamente alinhado, neste caso o mesmo acontece, como de pode verificar na parede onde há um ligeiro desvio dos planos. Por consequência, ao usar as matrizes de rotação e translação estimadas, ocorrem erros de estimativa, e um dos planos desalinhado é considerado como um objeto em movimento.

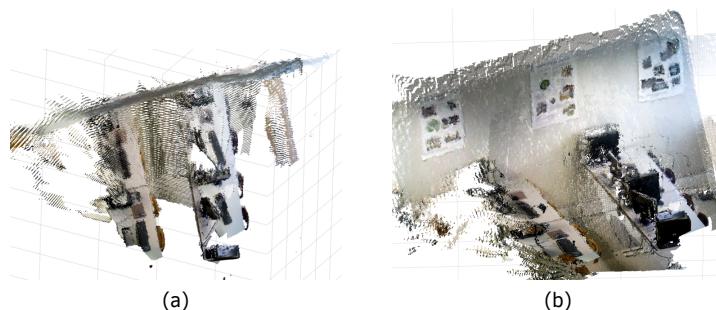


Figura 17: Fusão de nuvens de pontos usando o "dataset" **lab_piv_short**

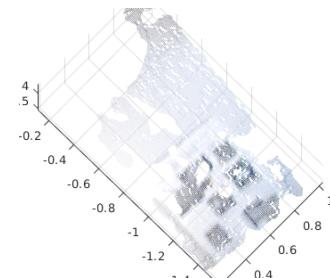


Figura 18: Objeto detetado em **lab_piv_short**

Aplicação do algoritmo ao dataset office

Neste dataset, pode-se verificar que o algoritmo não faz a fusão das várias nuvens de ponto ao contrário do que era expectável. Tal deve-se ao facto de não se encontrarem poucas "features" comuns entre imagens adjacentes (nomeadamente 2 e 3), sendo que nas poucas que são encontradas, a grande maioria são "outliers" que não são removidos pelo RANSAC.

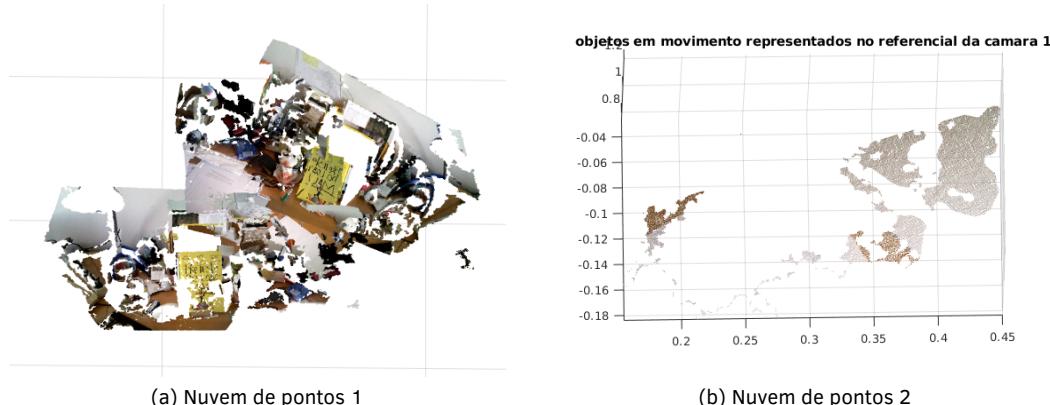


Figura 19

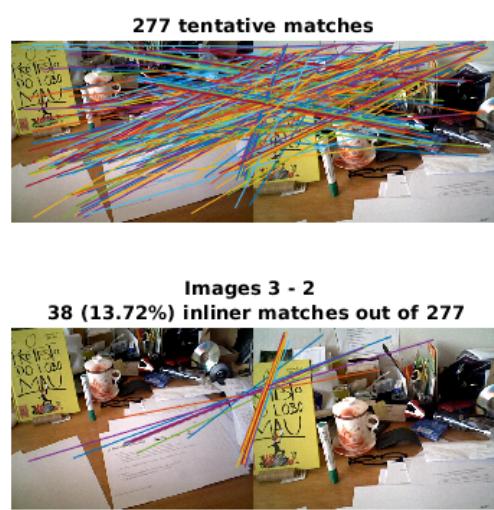


Figura 20

ANEXO B

Cálculo da matriz de rotação e translação entre pontos correspondentes

Neste anexo encontra-se a dedução que permite calcular uma transformação rígida entre dois conjuntos de pontos correspondentes. Mais uma vez, é reforçado que esta demonstração foi adaptada quase na sua integra de [3].

Assumindo que existem dois conjuntos de pontos $\{p_i\}$ e $\{p'_i\}$, sendo $i = 1, \dots, N$ sendo N o número de pares de pontos, pode-se determinar as matrizes de rotação e de translação ótimas relacionam duas nuvens de pontos resolvendo o problema de Procrustes.

Seja então $\{p'_i\}$ dado por:

$$p'_i = Rp_i + \mathbf{T} \quad (12)$$

Como o objetivo é obter as matrizes de rotação R e de translação T tais que:

$$(R, T) = \operatorname{argmin}_{R, T} \sum_{i=1}^N w_i \|(Rp_i + \mathbf{T}) - p'_i\|^2 \quad (13)$$

sendo w_i o peso correspondente a cada par de pontos (neste caso consideramos todos os pontos com o mesmo peso). Tal como referido em [3], considerando $F(T) = \sum_{i=1}^N w_i \|(Rp_i + \mathbf{T}) - p'_i\|^2$, podemos obter a translação óptima, derivando F em ordem a T :

$$0 = \frac{\partial F}{\partial T} = \sum_{i=1}^N 2w_i(p'_i - (Rp_i + \mathbf{T})) = 2T \sum_{i=1}^N w_i + 2R \sum_{i=1}^N w_i p_i - 2 \sum_{i=1}^N w_i p'_i \quad (14)$$

Re-arranjando termos podemos definir:

$$\bar{p} = \frac{\sum_{i=1}^N w_i p_i}{\sum_{i=1}^N w_i} \text{ e } \bar{p}' = \frac{\sum_{i=1}^N w_i p'_i}{\sum_{i=1}^N w_i} \quad (15)$$

Substituindo na equação anterior obtém-se o vetor de translação T :

$$T = \bar{p}' - R\bar{p} \quad (16)$$

Substituindo a equação anterior anterior na função que se pretende minimizar obtém-se:

$$\sum_{i=1}^N w_i \|(Rp_i + T) - p'_i\|^2 = \sum_{i=1}^N w_i \|(Rp_i + \bar{p}' - R\bar{p} - p'_i)\|^2 = \sum_{i=1}^N w_i \|R(p_i - \bar{p}) - (p'_i - \bar{p}')\|^2 \quad (17)$$

Pode-se então calcular a matriz de rotação R considerando que a translação é zero, e problema que vamos minimizar passa a ser dado por:

$$R = \operatorname{argmin}_R \sum_{i=1}^N w_i \|Rx_i - y_i\|^2, \text{ sendo } x_i := p_i - \bar{p} \text{ e } y_i := p'_i - \bar{p}' \quad (18)$$

Começando por simplificar a expressão a minimizar, obtém-se:

$$\begin{aligned} \|Rx_i - y_i\|^2 &= (Rx_i - y_i)^T (Rx_i - y_i) = x_i^T x_i - y_i^T Rx_i - x_i^T R^T y_i + y_i^T y_i \\ &= x_i^T x_i - 2y_i^T Rx_i + y_i^T y_i \end{aligned} \quad (19)$$

Substituindo na função a minimizar obtem-se:

$$\begin{aligned} \operatorname{argmin}_R \sum_{i=1}^N w_i \|Rx_i - y_i\|^2 &= \operatorname{argmin}_R \sum_{i=1}^N w_i (x_i^T x_i - 2y_i^T Rx_i + y_i^T y_i) \\ &= \operatorname{argmin}_R \left(\sum_{i=1}^N w_i x_i^T x_i - 2 \sum_{i=1}^N w_i y_i^T Rx_i + \sum_{i=1}^N w_i y_i^T y_i \right) \\ &\Rightarrow \operatorname{argmin}_R \sum_{i=1}^N w_i y_i^T Rx_i \end{aligned} \quad (20)$$

Esta última simplificação é feita, pois os restantes termos não dependem de R . Da mesma forma, o valor -2 é apenas uma constante, pelo que pode ser removida, simplificando a expressão a minimizar.

Seja então:

$$\sum_{i=1}^N w_i y_i^T R x_i = \text{tr}(W Y^T R X) \quad (21)$$

E a matriz de covariâncias S dada por $S = X W Y^T$. Então recorrendo à decomposição em valores singulares pode-se expressar $S = U \Sigma V^T$. Substituindo a decomposição em valores singulares na expressão do trasso, obtém-se:

$$\text{tr}(W Y^T R X) = \text{tr}(R X W Y^T) = \text{tr}(R U \Sigma V^T) = \text{tr}(\Sigma V^T R U) \quad (22)$$

Como as matrizes V , R e U são ortogonais, também $M = V^T R U$. Sabe-se ainda que as entradas de M (m_{ij}) respeitam:

$$1 = m_j^T m_j = \sum_{i=1}^d m_{ij}^2 \Rightarrow m_{ij}^2 \leq 1 \Rightarrow |m_{ij}| \leq 1 \quad (23)$$

Então o valor do traço é máximo se para as entradas $m_{ii} = 1$. Como a matriz M é ortogonal, quer isto dizer que M teria de ser a matriz identidade, e por consequência:

$$M = V^T R U = I \Rightarrow V = R U \Rightarrow R = V U^T \quad (24)$$

Este processo permite obter a matriz ortogonal óptima, que pode conter reflexões para além das rotações. Como a matriz de rotação tem de verificar $\det(R) = 1$, então quando $\det(R) = \det(V U^T) = -1$ significa que a matriz contém também uma reflexão. Uma forma geral de resolver este problema é dizer que a matriz de rotação vem dada por:

$$R = V \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & \det(V U^T) \end{bmatrix} U^T \quad (25)$$

Apesar deste último passo ser fácil de implementar em código, optou-se por multiplicar a última coluna de V por -1 quando a matriz de rotação calculada originalmente tem o seu determinante igual a -1.