INSTITUTO SUPERIOR TÉCNICO

MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES

PIV 2017/2018

Deteção e tracking de objectos utilizando câmaras de profundidade



Alunos / nº:

Mariana MARTINS, 80856

Filipe MARQUES, 73416

Ana Rita HILÁRIO, 81015

Grupo / Turno:

2ª Feira, 17h - 18h30m

Docente:

Prof. João Paulo COSTEIRA

23 de Dezembro de 2017

Conteúdo

1. Descrição do problema

O objetivo deste projeto é a deteção e rastreamento de um número variável de objetos em movimento num dado ambiente observado por duas câmaras estáticas. Para tal emprega-se o uso de Kinects, por forma a obter imagens RGB e de profundidade.

Dada esta situação, é necessário determinar a localização de um objecto de uma imagem (frame) para outra, em que o objecto se movimentou, para cada uma das câmaras. E, também, determinar que objectos correspodem a quais entre as mesmas, já que podem haver objectos apenas vistos por uma das câmaras.

Na primeira parte são dados os parâmentros extrínsecos (R e T) de cada uma das câmaras, na segunda parte estes parâmetros têm de ser descobertos. No final é suposto obter um vector de objectos, em que cada objecto contém 8 pontos com coordenadas X, Y e Z por cada frame em que foi detectado.

2. Resolução do problema

Para a implementação de uma solução para o problema proposto, propõe-se a seguinte divisão em subproblemas que está representada na figura 1:

1. Determinação dos parâmentros extrínsecos de cada câmara (para a parte 2)
2. Determinação do background visto por cada câmara
3. Para cada par de frames
   (a) Remover o background de cada frame, obtendo apenas o objecto (foreground) para cada câmara
   (b) Obter uma representação conjunta do foreground entre as duas câmaras (merged pointcloud) para a frame anterior e para a frame nova
   (c) Determinar a localização dos objectos (8 pontos nos extremos) através dessas representações para cada uma das frames
   (d) Determinar uma correspondência de pontos (match) entre os objectos de cada frame para ambas as câmaras
   (e) Descobrir a que objectos corresponde cada match para ambas as câmaras
   (f) Actualizar o vector de objectos

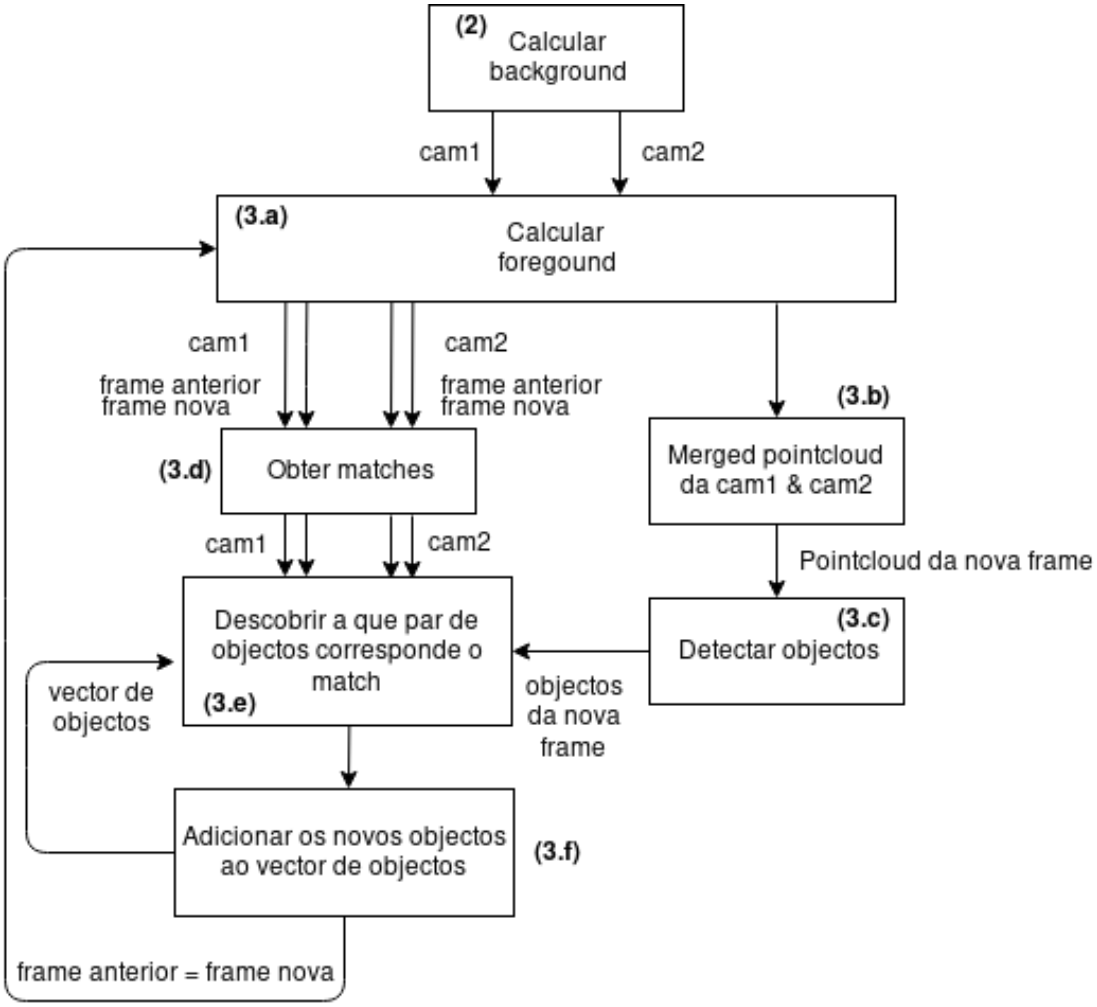

Figura 1: Funcionamento da solução implementada

2.1 Kinect

Estes sensores são capazes de captar informação de cor e de profundidade do ambiente sob observação, através do uso de uma camara de côr e uma câmara de profundidade, o que é essencial para a resolução do problema dado o modelo da câmara.

2.2 Remoção do background

2.2.1 Abordagem

Para determinar o background do ambiente visto, parte-se de um conjunto de imagens de profundidade e calcula-se a sua mediana.

Utiliza-se a mediana em vez da média, uma vez que a média permite a inclusão de "ruído". Para um determinado ponto, se este algumas vezes corresponder a um ponto num objecto, a média terá isso em conta no cálculo do background, enquanto que a mediana apenas terá em conta o ponto quando

corresponde a background, assumindo que mais de 50% das vezes esse ponto pertence ao background.
E opta-se pelas imagens de profundidade em vez de RGB, já que se o objecto tiver cores parecidas com a correspondente parte no background, maiores partes do objecto serão eliminadas do que calculado com as imagens de profundidade.
Toma-se um conjunto de no máximo 100 imagens para calcular a mediada, pois para os datasets existentes é suficiente para obter um background consistente e mais não melhoraria tanto para compensar o tempo de computação.

2.3 Cálculo do foreground

2.3.1 Abordagem

Para cada frame de cada câmara obtém-se o foreground em RGB, em profundidade e em 3D. O foreground é obtido pela subtração da imagem à de background em profundidade (calculada anteriormente). Na subtracção considera-se que se a diferença entre a profundidade do objecto e do background para um dado ponto é maior que 10cm, então é parte do foreground. Esta subtração gera uma imagem binária em que o foreground são pontos a 1.
A imagem binária é passada por um filtro bwareafilt, que extrai componentes que estão dentro de uma determinada range. Para este caso, definiu-se essa range como uma área de objecto entre 600 pixéis e a área total da imagem. Com a imagem resultante, aplica-se um filtro morfológico imopen, que faz erosão e dilatação dos pontos da imagem de acordo com a estrutura morfológico definida. Para este caso, definiu-se um quadrado com uma largura de 3 pixéis. Baseado na imagem binária final, calculam-se os ponto RGB, depth e 3D do foreground.

2.3.2 Resultados

Figura 2: Em RGB: Background (à esquerda) e primeira imagem do dataset maizena4 (à direita)

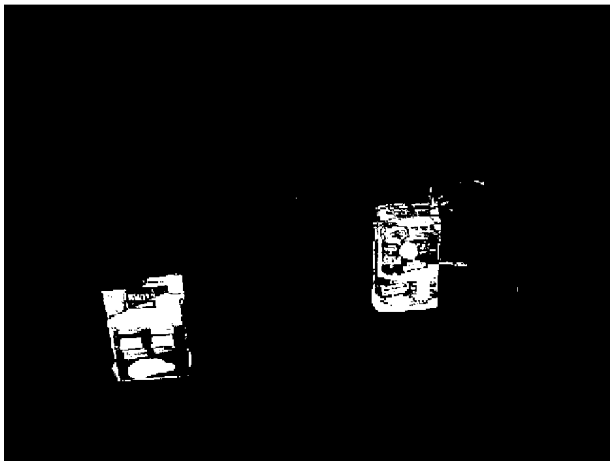

Figura 3: Foreground (da primeira imagem do dataset maizena4) em binário calculado com o backgroud definido em imagem RGB

Como dito na secção anterior, usando o background calculado em imagem RGB (figura 2), partes dos objectos que tinham cores parecidas ao background foram removidos (figura 3).

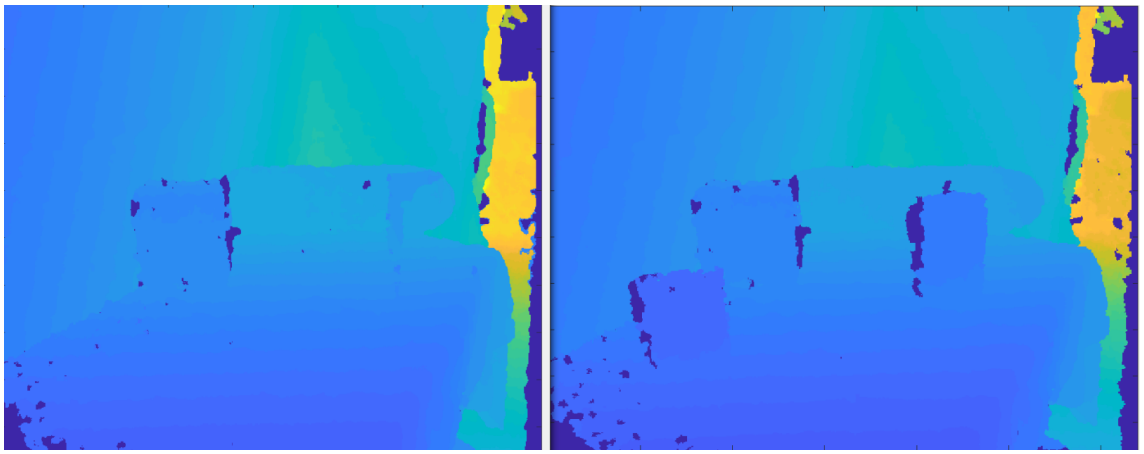

Figura 4: Em profundidade: Background (à esquerda) e primeira imagem do dataset maizena4 (à direita)

Figura 5: Foreground (da primeira imagem do dataset maizena4) em binário calculado com o backgroud definido em imagem de profundidade

Usando o background em profundidade (figura 4), o foreground (figura 5) contém muitos mais pontos do objecto.

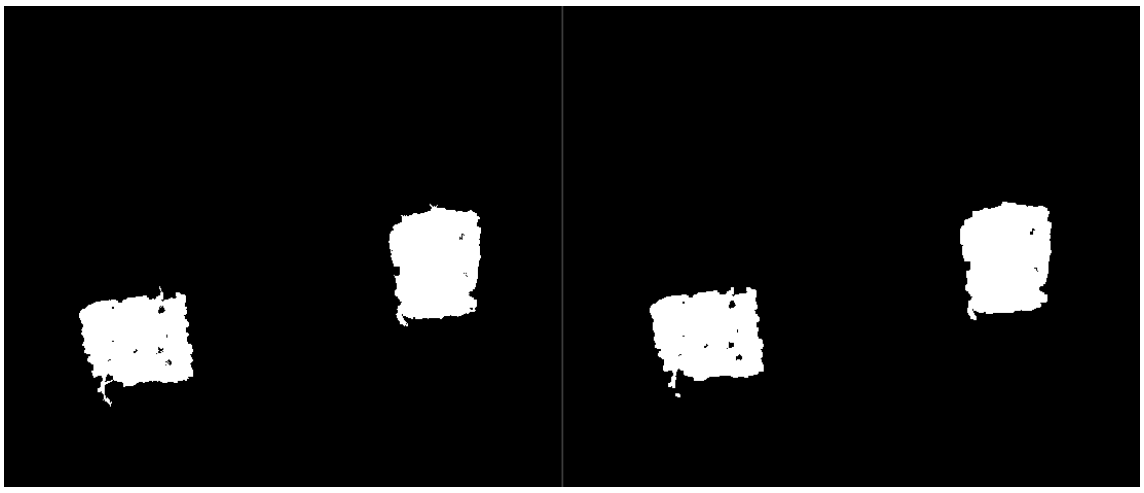

Figura 6: Foreground da figura 5 passado por bwareafilt (à esquerda) e depois pelo filtro morfológico imopen (à direita)

Pela figura 6, verifica-se que o filtro bwareafilt elimina todos os pontos que ficaram no foreground e não devem ser considerados um objecto. O filtro imopen torna os objectos mais regulares e dilata-os de forma a ficarem mais cheios, o que irá ajudar mais tarde quando se fizer a procura de pontos de interesse no objecto. Estes parâmetros não são os melhores para cada dataset em específico, mas são os que globalmente fazem com que todos tenham um foreground consistente.

2.4 Merged Pointcloud da Câmara 1 e 2

2.4.1 Abordagem

De maneira a facilitar detecção de quais objectos correspondem a quais entre as câmaras e a obtenção das 8 coordenadas por objecto, utiliza-se as imagens de foreground RGB, depth e 3D de cada câmara, obtidas no passo anterior, para gerar uma representação a três dimensões do foreground, pointcloud, por câmara. As 2 pointclouds são posteriormente unidas para obter uma representação de todo o espaço a ser analisado (que necessariamente tem todos os objectos). Esta merged pointcloud é gerada para cada frame, e pode ser observada na figura 8.

2.5 Detectar Objectos

2.5.1 Abordagem

Para detectar objectos diferentes é mais fácil pensar nos mesmos como um conjunto de pontos que não excedem uma certa distância máxima entre vizinhos. Como tal emprega-se o algoritmo 'Quick Union' para identificar grupos de pontos 3D que formam um objecto, considerando um número mínimo de pontos por forma a filtrar pontos outliers que não correspondem a nenhum objecto. Neste caso utilizou-se como distância máxima 20cm e como filtro 30 pontos.
Fazendo, então, a análise ponto-a-ponto da point cloud unida:

1. Para pontos que estejam a uma distância euclideana, um do outro, inferior a um valor pré-estabelecido, é lhes dado o mesmo identificador (label).

2. Os pontos obtidos são organizados por ordem do seu identificador (para diminuir o número de iterações) e posteriormente divididos em grupos de pontos com o mesmo identificador (isto é, em objectos).

3. Calcula-se as 8 coordenadas do objecto através dos valores máximo e mínimo de cada conjunto de pontos, para os 3 eixos X, Y e Z.

É formado um vector de objectos actuais, em que cada objecto contém as 8 coordenadas dos seus extremos, que vão ser usadas no próximo passo.

2.6 Obter Matches

2.6.1 Abordagem

Por forma a obter correspondências do foreground entre frames utiliza -se o algoritmo SIFT, Scale Invariant Feature Transform, capaz de detetar as características locais de uma imagem, keypoints, e gerar, para cada característica, uma vizinhança vetorial, com base na variação do gradiente na região em torno dessa característica, descriptors.

Os descriptors gerados são invariantes à rotação, translação e escalamento da imagem a que pertencem, sendo por isso utilizados para encontrar correspondências entre imagens consecutivas, através da ferramenta vl_ubcmatch, que recebe os descriptors para cada uma das imagens que se pretende comparar e um threshold de erro, até ao qual a falha na correspondência se considera aceitável.
Com os matches obtidos, converte-se para coordenadas do mundo real cada um dos keypoints que pertecem aos matches.
Este processo, é feito para a frame anterior e a actual, por cada câmara.

2.7 Correspondência entre objectos

2.7.1 Abordagem

Para descobrir a que par de objectos um match corresponde começa-se por definir uma matriz cujas linhas correspondem aos objectos da frame anterior, e as colunas aos objectos da frame actual. Esta matriz é utilizada para guardar a quantidade de matches (scores) que cada objecto da frame anterior tem com cada objecto da frame actual.
Para calcular então os valores desta matriz, a cada match verifica-se em que objecto o keypoint desse match da frame anterior está incluido,e posteriormente em que objecto da frame actual esse keypoint está contido, com os indíces dos objectos obtidos passa-se a incrementar o elemento da matriz de correspondências.
Novamente este processo é repetido para ambas as câmaras a serem consideradas, somando à matriz de correspondências.

2.8 Actualização do vector de objectos

2.8.1 Abordagem

Tendo em conta a matriz de correspondências obtida no passo anterior e o vector de objectos, passa-se a fazer a sua actualização.
Esta actualização é feita de modo iterativo, sendo que a cada iteração é procurado o elemento do valor máximo da matriz de correspondencias e extraídos os seu indíces, que sao utilizados para pôr as 8 coordenadas novo objecto no sítio correspondente do vector de objectos.
Se permanecerem objectos que não foram correspondidos aos da frame anteriror, são adicionados como novos objectos no vector de objectos final.

2.9 Determinação de parâmentros extrínsecos

2.9.1 Abordagem

Inicialmente nesta parte, implementou-se o método RANSAC, que é um processo iterativo utilizado para estimar os parâmetros de um modelo matemático a partir de um conjunto de dados adquiridos que contém outliers e inliers, dados que, respetivamente, não são descritos pelo modelo considerado e outros que embora possam estar sujeitos a ruído se enquadram no modelo a menos de um valor de erro definido. Assume-se ainda que o conjunto de dados a ser utilizado contém mais inliers do que outliers, se acontecer o contrário é impossível estimar um modelo adequado.
O objetivo é obter a matriz de rotação R e o vetor de translação T entre a câmara 2 e câmara 1, assumindo que o referencial do mundo coincide com o desta última.

Para adaptar este método ao problema, definiu-se o seguinte:

1. O algoritmo SIFT calcula os keypoints e os matches entre câmaras que irão ser utilizados como possíveis inliers. É de notar que se elimina matches cuja profundidade dos keypoints é nula.

2. Geram-se 4 indíces aleatórios que vao corresponder a matches (estes são os inliers iniciais). 4 é o número mínimo necessário para a estimação do modelo em causa.

3. Usando o Procrustes com os 4 matches aleatórios obtém-se uma matriz de rotação e um vector de translação - o modelo.

4. Todos os matches que não os 4 definidos em (2), são passado pelo modelo gerado em (3) e se possuirem um erro menor do que o *threshold* estabelecido, guardam-se.

5. Calcula-se o modelo novamente com os 4 inliers iniciais e os obtidos em (4) e verifica-se se é o melhor modelo.

6. Repetem-se os 4 pontos anteriores para N iterações, guardando-se sempre o melhor modelo.

Para este caso, utilizaram-se 500 iterações, o erro oferecido pela função procrustes (do mathworks) para determinar se é o melhor modelo e um threshold de 1cm.

2.9.2 Resultados

Implementando a abordagem descrita a cima, verificou-se que o erro estava na ordem dos milímetros. Uma forma de perceber se este erro é aceitável ou não, é observar a merged pointcloud das duas câmaras. Como se vê na figura 7, este erro é inaceitável.

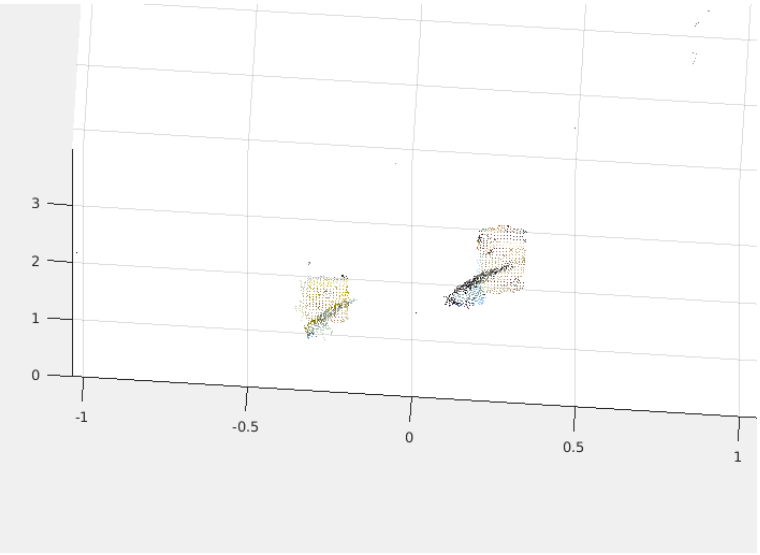

Figura 7: Pointcloud com o RANSAC implementado como suposto

Como tal, implementou-se um método alternativo:

1. Define-se um conjunto máximo de M imagens distribuídas igualmente pelo dataset (de forma a garantir que uma imagem específica do dataset não impede o cálculo de parâmetros extrínsecos que seja aceitáveis)
2. Por cada imagem, tal como anteriormente, usa-se o SIFT para determinar os matches.
3. Definem 4 matches aleatórios.
4. Calcula-se R e T pelo procrustes com esses matches, e guarda-se esse modelo caso seja o melhor, com base no erro oferecido pela função procrustes como anteriormente.
5. Repetem-se os pontos (3) e (4) por N iterções.
6. Repete-se o método para as restantes imagens.

Analisando o erro deste modelo, observa-se que esta na ordem dos micrómetros. E a pointcloud é muito mais fiável como se observa na figura 8. Usou-se 7 para o número máximo de imagens a analisar, dado o tempo de computação.

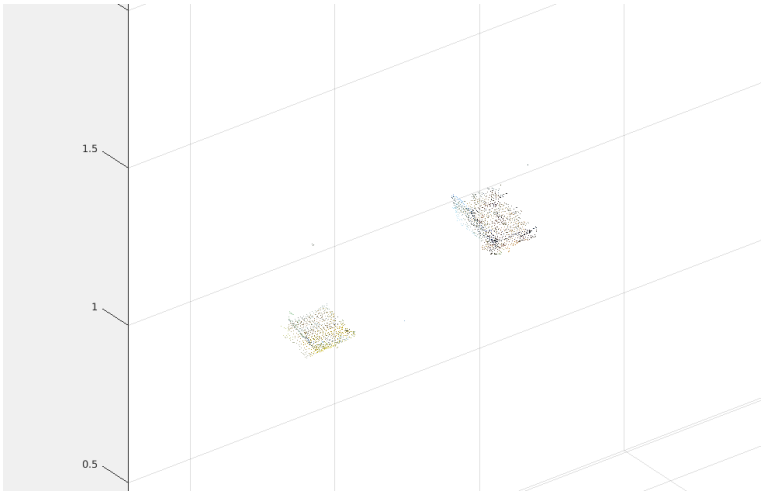

Figura 8: Pointcloud com o método alternativo

O problema deste método é que, por ser baseado em apenas 4 pontos, não é muito estável e erro torna-se muito maior que no método anterior.
O método original pode ser consultado em https://github.com/Mrrvm/Computer-Vision/bin.

2.9.3 Limitações

Determinar os parâmetros extrínsecos baseia-se na determinação de matches de keypoints entre as câmaras, pelo que tem de haver features parecidas entre as duas imagens.
No dataset corredor1 é quase impossível determinar R e T dado que uma câmara está em oposição com a outra, e como tal as features parecidas baseiam-se nas paredes (que têm poucos pontos distintos) e objectos cuja superfície esteja paralela à visão da câmara.
Datasets que tenham objectos com cores iguais como o stillnature, também impedem uma determinação consistente de R e T, já que objectos diferentes podem ter matches.

3. Conclusão

A principal dificuldade encontrada ao longo realização do projecto teve a ver com a influência dos efeitos estocásticos nos mecanismos de processamento, que geram problemas nos métodos de detecção e correspondência.

Apesar de serem necessários mais testes para garantir a capacidade de generalização do projecto implementado, consideramos que a capacidade de detecção e associação da presente implementação está bem desenvolvida, sendo que os testes feitos mantiveram o correcto número de objectos.

Uma limitação encontrada, está relacionada com o desaparecimento de objectos de uma frame para a seguinte, sendo que em frames seguintes, no caso do reaparecimento do mesmo objecto, este é tratado como um novo objecto e adicionado ao vector de objectos como tal. Uma forma de resolver este problema seria, sacrificando a eficiência do programa, aquando da detecção de um objecto que não corresponde a nenhum objecto da frame anterior, processar os passos de detecção e associação entre a frame actual e as ultimas frames em que os objectos que não estavam presentes na frame anterior aparecem. Prevê-se que este procedimento poderia oferecer uma solução para o problema apresentado, mas associado estaria um aumento de complexidade considerável que não parece valer a pena.

Adicionalmente, uma outra implementação para o projecto foi tentada, em que após a união das point clouds das duas câmaras para uma frame, se processava uma projecção desta união para 2D, e era com esta imagem em que se procuravam as matches entre frames, para oferecer correspondencia entre objectos. No entanto esta implementação provou conduzir à perda de keypoints importantes, resultando numa associação inferior que inclusivamente levava mais tempo de computação. Esta implementação pode ser consultada em https://github.com/Mrrvm/Computer-Vision/bin/main2.

Outras sugestões de melhoria consideradas passam pelo uso de gradientes de movimento, em métodos iterativos que permitirião auxiliar a detecção de objectos com base na predição de trajectória, mas não só esta sugestão foge ao objectivo do projecto como, só seria util para datasets com um numero elevado de medições.

No final concluimos também que o uso de duas câmaras de profundidade para detecção e rastreamento, fornecem uma quantidade de informação mais favorável ao processo uma vez que 2 perspectivas de objectos em movimento dificultam a suprecção de um deles quando as suas trajectórias se cruzam, assim como diminui o erro de precisão. Por fim, é de notar que a detecção de objectos tem muitas limitações e é extremamente difícil ajustar o programa para servir todos os datasets.

4. Anotações

1. Na parte 1 foi considerado que os parâmetros extrínsecos da câmara 1 são sempre a matriz identidade para a rotação e 0 para a translação. Pelo que se forem dados parâmentros que não esse no argumento cam1toW não vai funcionar bem. No entanto, a correcção seria simples e passaria por passar os pontos da câmara 2 pelos pârametros extrínsecos dela e pelos pâramentros extrínsecos da câmara 1 inversos.

2. O código adicional está disponível em https://github.com/Mrrvm/Computer-Vision/