

Mestrado Integrado em Engenharia Electrotécnica e de Computadores

**Processamento de Imagem e Visão**

1º Semestre – 2016/2017

**Relatório do Projeto**

3D Reconstruction

Turno 2ª-feira 17 horas

73177 – Ana Catarina Gonçalves

78918 – Hugo Conde Barroso

79007 – Tomás Miguel Cardoso

Prof. João Paulo Costeira

**Data de Entrega:** 23/12/2016

**Índice**

|  |  |  |  |
| --- | --- | --- | --- |
| * 1. Introdução |  |  | 1 |
| * 1. Part I – 3D scene “scanning” |  |  | 2 |
| * 1. Descrição do Problema |  |  | 2 |
| * 1. Solução Implementada |  |  | 2 |
| * 1. Argumentos da Função |  |  | 3 |
| 1. Leitura das Imagens |  |  | 4 |
| * 1. Image Features |  |  | 4 |
| * 1. Matching Features |  |  | 5 |
| * 1. Transformação de Corpo Rígido |  |  | 5 |
| * 1. Reconstrução do cenário 3D |  |  | 7 |
| * 1. Conclusão |  |  | 8 |
|  |  |  |  |

1. **Introdução**

O projeto tem como objetivo principal a aplicação da matéria teórica lecionada nas aulas, mais concretamente ao processamento de imagem.

São explorados os tópicos de reconstrução de uma imagem 3D a partir de imagens 2D com recurso a transformações de corpo rígido e utilização de algoritmos como SIFT e o RANSAC.

O projeto foi realizado em Matlab e está estruturado numa única função para cada parte. De seguida, é descrito o procedimento realizado para resolução de ambos os problemas.

1. **Parte I – 3D scene "scanning" with the Kinect images**
   1. **Descrição do Problema**

Pretende-se com esta parte do projeto estudar e desenvolver soluções para o problema da localização e mapeamento simultâneo. Este problema aparece muito naturalmente, no ramo da robótica, com a necessidade de fornecer a um sistema robótico a capacidade de reconhecimento autónomo do ambiente em que se encontra inserido e de se localizar no mesmo.

Existem atualmente diversas formas de resolver este tipo de problema, a escolha entre as possíveis soluções depende essencialmente do conjunto de sensores disponíveis para o efeito.

Neste trabalho, assumiu-se que o robot para o qual estaríamos a desenvolver um sistema de auto-localização utiliza como sensor uma câmara Kinect, que é composta por uma câmara RGB e uma câmara de profundidade (baseada em raios infravermelhos). É, por isso, possível obter a informação associada à cor e à localização tridimensional, no referencial da câmara Kinect, de todos os pontos visualizados por esta, o que permite a construção de uma point cloud.

Assim, o objetivo deste trabalho é conjugar as imagens recolhidas nas câmaras de modo a construir as respetivas point clouds, as transformações entre elas e propagar as mesmas de tal forma a que as point clouds se situem numa única frame final de referência, reconstituindo assim um cenário 3D.

* 1. **Solução Implementada**

A relação entre pontos tridimensionais reais e pontos bidimensionais no plano de imagem é obtida através do modelo da câmara, definido em coordenadas homogéneas, da seguinte forma,

em que,

* corresponde a um fator de escala;
* corresponde às coordenadas homogéneas no referencial do plano da câmara;
* K é uma matriz 3 x 3 correspondente aos parâmetros intrínsecos da câmara;
* [R | t] é uma matriz 3 x 4 correspondente aos parâmetros extrínsecos (posição) da câmara;
* corresponde às coordenadas homogéneas no referencial "mundo" predefinido;
* P = K[R | t] é uma matriz 3 x 4 correspondente à matriz da câmara.

Aplicando este modelo à câmara RGB estática e tendo em conta que os parâmetros intrínsecos da mesma são conhecidos, a sua posição pode ser estimada à custa de um conjunto de pontos . Os pontos aqui considerados correspondem às coordenadas, no plano da câmara RGB estática.

De acordo com o modelo referido, a posição da câmara RBG estática corresponde à matriz da câmara P. Formalmente, a matriz P representa a posição da câmara RGB relativamente ao referencial "mundo", no entanto e neste caso especifico, a câmara RGB é estática e o referencial "mundo" é aquele que sofre o deslocamento ao longo do tempo. Nestas condições, uma interpretação igualmente correta e mais intuitiva seria a de que, a matriz P corresponde à posição do objeto de calibração relativamente ao referencial da câmara RGB estática.

A estimativa da matriz P depende da orientação do objeto de calibração relativamente à câmara RGB estática.

* + 1. **Argumentos da Função**
* *image\_names*, um array de estruturas que contêm os nomes das imagens. Cada elemento do array é uma estrutura do tipo image\_name(k).depth, uma string com o caminho até ao ficheiro .mat que contém a informação sobre a profundidade da imagem k, ou image\_name(k).rgb, também uma string com o caminho até a um ficheiro de extensão .jpeg ou .png, ou seja, a imagem RGB;
* *K\_depth,* uma matriz 3x3 que contém os parâmetros intrínsecos da câmara de profundidade;
* *K\_rgb,* uma matriz 3x3 que contém os parâmetros intrínsecos da câmara RGB;
* *Rdtrgb,* a matriz de rotação do referencial da câmara de profundidade para o referencial da câmara RGB;
* *Tdtrgb,* a matriz de translação do referencial da câmara de profundidade para o referencial da câmara RGB.
  + 1. **Leitura das Imagens**

Tal como foi descrito anteriormente, é passado como argumentos à função uma estrutura com os full paths para os ficheiros, tanto das imagens png e .mat com a informação sobre a depth de cada imagem. De modo a ler e armazenar a informação sobre todas as imagens, estas estruturas foram percorridas e foram obtidas imagens em escala cinzenta, tamanho de cada imagem e point Clouds e cores rgb das mesmas, bem como as coordenadas espaciais (x,y,z) de cada pixel de cada imagem. Toda esta informação foi armazenada em duas estruturas, uma retratando tudo relativamente à componente 2D da informação e outra para a componente 3D.

Para calcular as coordenadas espaciais foi usada a função disponibilizada pelo professor, get\_xyzasus, que faz uso da informação sobre a depth de cada imagem e da matriz de parâmetros intrínsecos da câmara de profundidade da kinetic de modo a calcular as coordenadas espaciais.

De modo a calcular a point cloud de cada imagem, usou-se a função pointCloud da biblioteca computer vision do Matlab e para ober a informação sobre a cor da imagem, é usada novamente uma função disponibilizada pelo professor, get\_rgbd.

* + 1. **Image Features**

De modo a transpor as point clouds de cada imagem, é primeiro necessário encontrar os pontos que servirão como base para fazer a ligação entre um par de imagens. Pontos estes que são os pontos-chave da imagem, como por exemplo cantos, arestas, mudanças de cor e luminosidade, entre outros. Estes pontos são denominados de features e são obtidos usando a a função vl\_sift que implementa o algoritmo SIFT (Scale Invariant Feature Transformation). Esta função encontra-se presente na toolbox VL Feat. Esta função recebe com parâmetro uma imagem em escala cinzenta e retorna uma matriz as coordenadas do centro de cada feature frame (coordenadas da feature na imagem) e a escala e orientação da frame, mas estes últimos dois valores são irrelevantes no âmbito deste projeto. A função retorna também um vetor que serve como descriptor de cada feature.

* + 1. **Matching Features**

Agora que se tem as features de cada imagem, é necessário fazer um matching das mesmas entre cada imagem, para obtermos os pontos comuns entre duas imagens e calcular a transformação entre estes.

Usou-se a função vl\_ubcmatch, também do VL Feat toolbox. Esta função recebe como argumentos os descriptors, obtidos com o vl\_sift, das duas imagens a comparar e retorna uma matriz com os matches e um vetor com os pesos de cada match, ou seja, a quantização do quão bom é um match.

Assumindo que as imagens a comparar são as imagens 1 e 2, para cada feature da imagem 1, o vl\_ubcmatch calcula a distância euclidiana do descriptor correspondente a essa feature a todos os descriptors da imagem 2 e minimiza esta função, ou seja, obtém o feature da imagem 2 cujo descriptor está a menor distância do da imagem 1 a apurar. Esta distância é ainda comparada com um treshold, que traduz uma distância mínima para a qual é satisfeita a condição de match. Se a distância euclidiana entre os dois descriptors encontrados for menor que este treshold, então ocorre um match.

* + 1. **Transformação de Corpo Rígido**

Para calcular a matriz de transformação de corpo rígido a partir das correspondências efetuadas no ponto anterior proceder-se-á do seguinte modo:

* Cálculo dos centroides do conjunto de pontos de cada imagem

Para tal basta calcular a média de todos os pontos considerados para cada imagem

Onde N é o número total de pontos considerados para uma dada imagem e os pontos considerados da mesma imagem.

* Cálculo da matriz de rotação

Utilizou-se a decomposição em valores singulares (SVD) para este passo. Mas primeiro é necessário centrar os pontos para que os centroides se encontrem na origem, removendo assim a componente de translação e facilitando o cálculo da matriz de rotação.

Para calcular então a matriz de rotação usou-se a função svd, que faz a decomposição em valor singular da entrada, produzindo uma matriz diagonal S e matrizes unitárias U e V, de modo a que H = U\*S\*V‘ . Como entrada da função svd, foi passada a multiplicação das point clouds das imagens cuja transformação se quer calcular

Sendo R a matriz de rotação.

* Cálculo da matriz de translação

Por fim, o vetor de translação é dado por

Num problema como este, em que as correspondências entre pontos não são sempre as corretas, existe uma grande probabilidade de existirem outliers, o que leva a uma má estimação destas matrizes.

Uma alternativa ainda melhor do que a apresentada seria utilizar o RANSAC, um método utilizado para estimar parâmetros de um modelo quando se observam outliers.

Este algoritmo recebe um conjunto de pontos observados e um modelo cujos parâmetros se pretende estimar e utiliza várias combinações dos pontos mencionados para estimar, neste caso, a matriz de rotação e translação. Depois, tendo em conta um dado valor de threshold, após várias iterações (combinações de pontos), verifica qual foi a combinações de pontos que originou mais inliers, ou seja, após estimação da matriz R e T, verifica quantos pontos de uma imagem se aproxima dos correspondentes pontos anteriormente especulados, e que tenham um erro inferior ao threshold.

No entanto, não foi possível implementar tal algoritmo, pois as dificuldades encontradas na implementação do mesmo não foram devidamente ultrapassadas.

* + 1. **Reconstrução do cenário 3D**

Para a reconstrução do cenário 3D, são feitas transformações sucessivas de imagens consecutivas para a imagem escolhida como referência para o world frame através da equação:

Onde P são as coordenadas espaciais da imagem a transformar, R a matriz de rotação entre o referencial das duas imagens, T o vetor de translação entre os referenciais das duas imagens e P’ as coordenadas espaciais de P, mas no novo referencial. Aplicando a este P’ novamente a equação anterior, mas usando a matriz de rotação e vetor de translação para a imagem seguinte, obtendo-se P’’, que são as coordenadas espácias P no referencial da imagem seguinte e assim sucessivamente até cada imagem ser deslocada para o referencial de referência.

Obtém-se assim N matrizes P que traduzem as coordenadas espaciais de cada pixel das N imagens utilizadas, todas elas calculadas para o mesmo referencial.

1. **Conclusão**

Como se pode observar pela imagem 1, construída usando o data set newpiv4, conclui-se que a solução proposta apresenta resultados plausíveis, o que demonstra que o método utilizado funciona, ainda que incompleto. Tal deve-se ao facto das correspondências feitas entre as diferentes imagens não terem sido corretamente efetuadas, permitindo a existência de algumas correspondências erradas que influenciam claramente o cálculo das matrizes de rotação e translação. Como já referido, isso poderia ser resolvido com a implementação do RANSAC.

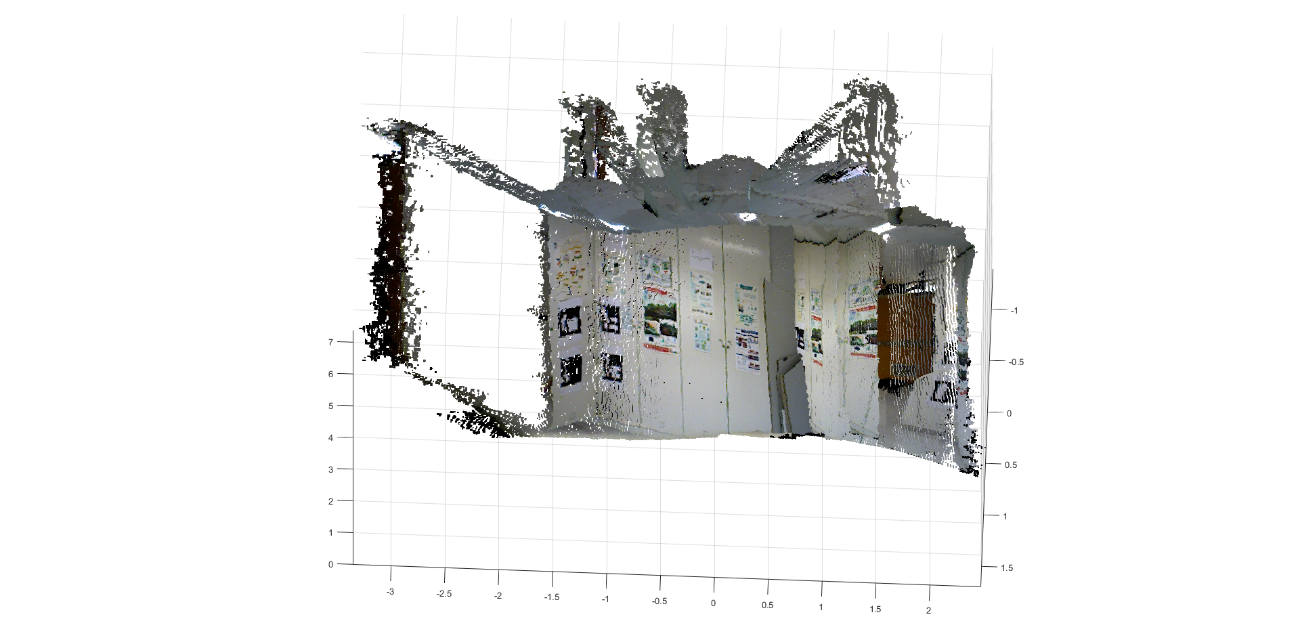


Figura 1 - Reconstrução de cenário 3D.