UNIVERSIDADE D
COIMBRA

MESTRADO EM ENGENHARIA E DE COMPUTADORES

Relatório de Visão por Computador

3D Reconstruction

Duarte de Sousa Cruz, 2017264087

Universidade de Coimbra
1 de Junho de 2022

1 Introduction

Neste trabalho é nos pedido para recuperar a estrutura 3D de um ambiente, juntamente com a posição da câmara, tendo apenas duas perspetivas do mesmo. Para isso, vão ser usados dois algoritmos:

1. Sparse Reconstruction
2. Dense Reconstruction

Técnicas de reconstrução 3D são usadas em campos como a ciência, medicina , robôs móveis ou realidade aumentada.

No relatório vamos explicar a implementação dos três algoritmos propostos e apresentar os resultados obtidos.

As imagens usadas para a reconstrução 3D foram as seguintes,

Figure 1: Imagens usadas para a reconstrução 3D

2 Sparse reconstruction

2.1 Fundamental Matrix

A matriz fundamental relaciona os pixeis de uma câmara para os pixeis de outra. A restrição entre estes pontos pode ser relacionada como restrição epipolar.

Para fazer o cálculo desta matriz foi usada a função *computeF* que vai ter como parâmetros de entrada os pontos da imagem da direita e os mesmo pontos na imagem da esquerda.

Vamos começar por normalizar os pontos das imagens fornecidas, usando a função *normalization* desenvolvida no trabalho anterior.

Após a normalização dos pontos vamos calcular a matriz A dada por:

$$\begin{bmatrix} x_1x'_1 & y_1x'_1 & x'_1 & x_1y'_1 & y_1y'_1 & y'_1 & x_1 & y_1 & 1 \\ x_2x'_2 & y_2x'_2 & x'_2 & x_2y'_2 & y_2y'_2 & y'_2 & x_2 & y_2 & 1 \\ \dots & & & & & & & & \end{bmatrix} \tag{1}$$

sendo (x, y) os pontos da primeira imagem e os (x', y') os pontos da segunda.

Obtendo esta matriz vamos ter de fazer a sua decomposição em valores singulares, $A = UDV^T$, sendo F a última coluna de V. Como temos de forçar F a ser de rank 2, vamos decompor a matriz F em valores singulares e substituir o menor valor singular da diagonal da matriz D por zero, ficando:

$$F = UDV^T \tag{2}$$

Para otimizar a nossa solução vamos usar a *normalized eight point algorithm* e utilizar a função *refineF* fornecida pelo professor.

$$Fd = T_r^T F T_l \tag{3}$$

sendo T_r e T_l as matrizes transformação obtidas após a desnormalização dos pontos das respetivas imagens.

As linhas epipolares obtidas através desta matriz podem ser visualizadas abaixo.

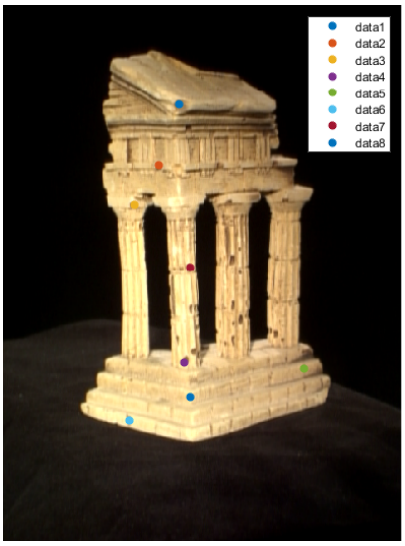

Figure 2: Pontos seleccionados

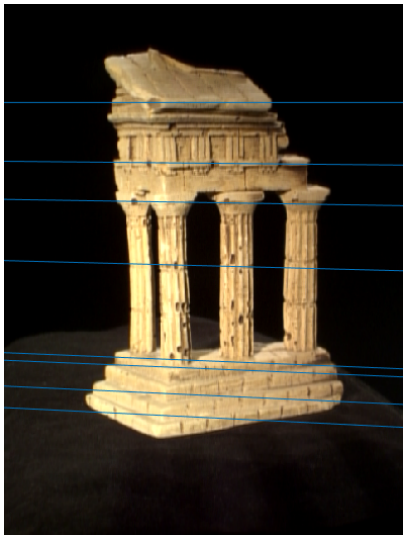

Figure 3: Linhas obtidas

2.2 Correpondências epipolares

No lab2 trabaalhamos com *feature detectors* and *deature descriptors* e o uso para encontrar pontos em commum entre duas imagens. O uso da matriz fundamental é uma boa ajuda para a tarefa, pois define a linha noutra imagem onde a correspondência para um ponto está localizado.

Começamos por calcular os parâmetros da linha epipolar correspondente a um ponto (x, y). Sendo a linha epipolar parametrizada da forma:

$$l \Rightarrow ax + by - c = 0 \tag{4}$$

a linha epipolar l vai ser igual a:

$$l = Fx = l = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{5}$$

e normalizada de acordo com a sua norma, $|l| = \sqrt{a^2 + b^2}$.

Por último, vamos usar os descritores para achar os pontos em comum. O descritor usado foi o *Simple* e o critério de matching usado foi o *ratio*.

```
im1=double(rgb2gray(im1));

[DesciptorsImg1]=SimpleFeatureDescriptor(im1,pts1,15);
pts2=[];

for n=1:size(l,1)

    for x=1:size(im1,2)
        p2(x,2) = -(l(n,1) * x + l(n,3))/l(n,2);
        p2(x,1)=x;
```

```
    end

    [DesciptorsImg2]=SimpleFeatureDescriptor(im2,p2,15);
    [Match]=RatioFeatureMatcher(DesciptorsImg1(n,:),DesciptorsImg2);
    pts2=[pts2;Match(2) floor(p2(Match(2),2))];
end
```

Figure 4: Correspondências epipolares

2.3 Matriz Essencial

Conhecida a matriz fundamental e as matrizes intrínsecas da câmara, a estimação da matriz essencial torna-se bastante simples. Calcula-se essa matriz aplicando a seguinte fórmula:

$$E = k_2^T F K_1 \tag{6}$$

sendo K_1 e K_2 as matrizes intrínsecas da câmara para as respetivas imagens.

2.4 Estrutura 3D esparsa

A estimação dos pontos 3D foi implementada usando uma função dada pelo professor. Essa função tem como entrada os pontos da imagem direita e correspondências da esquerda e as matrizes de projeção de ambas as imagens.

Antes de chamarmos a função temos que calcular P_1 e P_2. Para o cálculo de P_1 usámos a função:

$$P_1 = K_1 * [I|0] \tag{7}$$

sendo K_1 a matriz dos parâmetros extrínsecos da câmara 1. Para o de P_2 foi utilizada uma função *camera2* fornecida pelo professor. No entanto, esta função vai retornar quatro possibilidades para a matriz projeção.

A estrutura 3D vai ser calculada para as quatro possibilidades escolhendo depois aquela que produz uma estrutura 3D que esteja em frente a ambas as câmaras.

Para estimação dos pontos 3D vamos concatenar os pontos 2D de ambas as imagens e vamos resolver o seguinte sistema de equações lineares homogéneas:

$$\begin{bmatrix} yp_3^T - p_2^T \\ p_1^T - xp_3^T \\ y'p_3'^t - p_2'^T \\ p_1'^T - x'p_3'^T \end{bmatrix} X = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{8}$$

sendo p a respetiva coluna da matriz de projeção 1 e p' da câmara 2 e (x, y), (x', y') o par de pontos correspondentes das imagens.

Para o cálculo deste sistema de equações foi feita a decomposição em valores singulares da primeira matriz.

Figure 5: Triangulação

3 Dense Reconstruction

3.1 Imagem Rectificada

O primeiro passo para a reconstrução densa é retificar as imagens esquerda e direita. O processo consiste na transformação das imagens para que estas se apresentem no mesmo plano.

Ao retificar as imagens vai facilitar bastante a correspondência de pontos entre as duas, pois vai restringir o espaço de procura. Com as duas imagens retificadas apenas variamos a coluna do píxel que comparamos na imagem 2.

A função *rectifyMatrices* retorna as matrizes M1 e M2 que são as matrizes de retificação das duas imagens.

```
% optical centres of the cameras
c1n=-inv(K1*R1)*K1*t1;
c2n=-inv(K2*R2)*K2*t2;

% new rotation matrix
r1=(c1n-c2n)/norm(c1n-c2n);
r2=cross(R1(3,:)',r1);
r3=cross(r1,r2);

R1n=[r1'; r2'/norm(r2); r3'/norm(r3)];
R2n=R1n;

% intrinsic parameters matrix k
K1n=K2;
K2n=K2;

t1n=-R1n*c1n;
t2n=-R2n*c2n;

% Retification matrices
M1=K1n*R1n*inv(K1*R1);
M2=K2n*R2n*inv(K2*R2);
```

Podemos ver na Fig. que as linhas epipolares são agora totalmente horizontais.

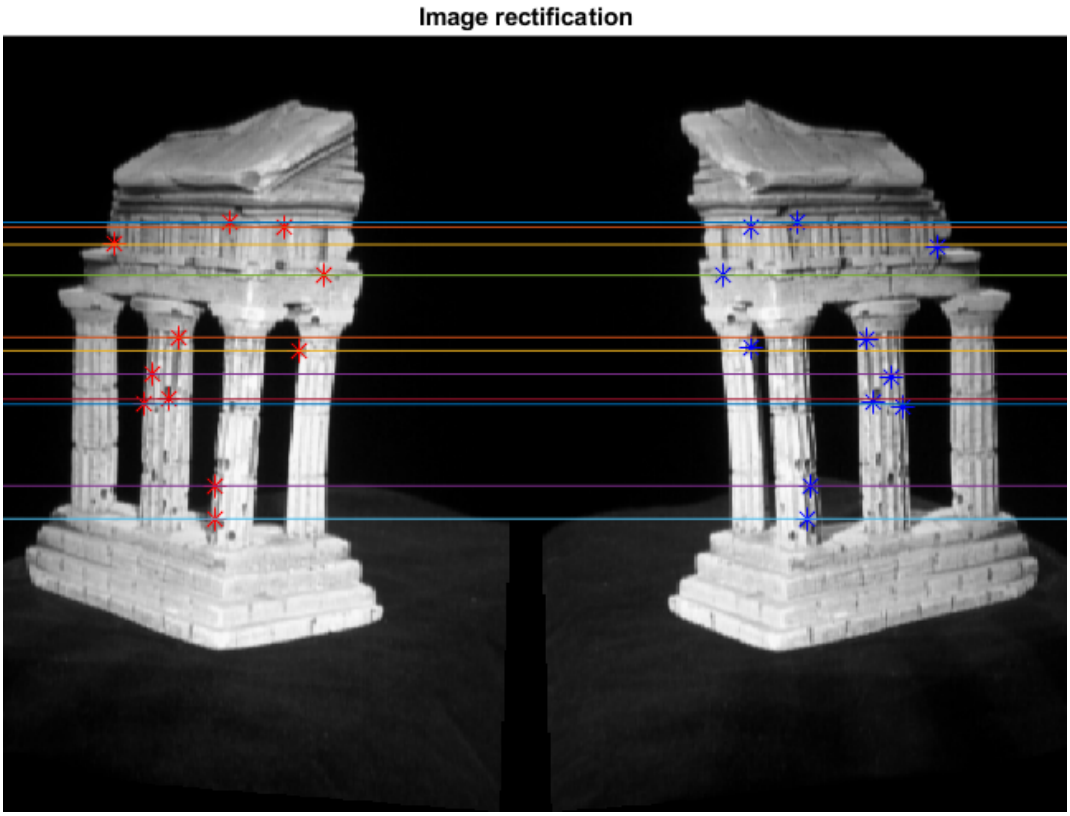

Figure 6: Imagens retificadas

3.2 Mapa de Disparidade

Disparidade é a distância entre a posição de dois pontos, onde a cor branca mais acentuada significa que estão mais perto um cor mais escura que estão mais longe.

Em primeiro vamos encontrar todos os descritores para cada píxel da imagem 1, para tal, vamos usar outra vez a função das labs anteriores *SimpleFeatureDescriptor*. Para forma de otimizar a função, vamos ignorar o fundo preto e só considerar os píxeis não nulos.

Com ambas as imagens retificadas, a procura dos respetivos descritos na imagem 2 vai ser pela seguinte fórmula:

$$dispMap(y,x) = \underset{0 \leq d \leq maxDisp}{argmin} \; dist(im1(y,x), im2(y,x-d)) \tag{9}$$

```
for i=1:size(pts1,1)
        p2 = [];
        Match = [];
        lowLimitX = max([pts1(i,1)-maxDisp 1]);
        highLimitX = min([pts1(i,1)+maxDisp size(im1,2)]);
        p2(:,1) = lowLimitX:highLimitX;
        p2(:,2) = y(i);

        [DesciptorsImg2]=SimpleFeatureDescriptor(im2,p2,windowSize);
        [Match,~]=SSDFeatureMatching(DesciptorsImg1(i,:),DesciptorsImg2,10);

         aux = min(Match(1,2:end-1));
         d = pts1(i,1) - p2(aux,1);
         dispM(y(i),x(i)) = abs(d);
end
```

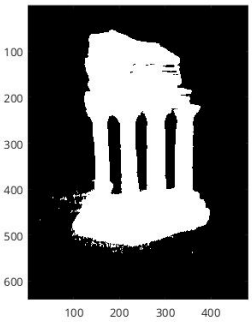

Figure 7: Mapa de Disparidade

3.3 Mapa de Profundidade

Conhecendo a matriz de disparidade podemos agora estimar a profundidade para cada píxel através da seguinte equação:

$$depthM(y,x) = \frac{bf}{dispM(y,x)} \tag{10}$$

sendo b a distância entre os centros ópticos e $f = K_1(1,1)$.

Para os pixeis onde a disparidade é zero a profundidade também deve ser.

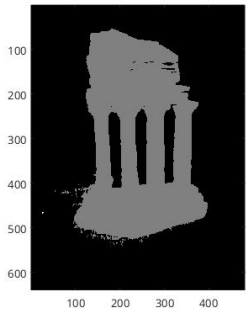

Figure 8: Mapa de Profundidade

4 Conclusão

Em conclusão, analisando os resultados apresentados, vemos que os algoritmos apresentados para a reconstrução esparsa teve o comportamento esperado. Mas, o algoritmo da densidade e disparsidade ficaram a não funcionar corretamente.