Project 4 Writeup

In the beginning...

A calibração de câmera geométrica, também conhecida como ressecção de câmera, determina os parâmetros de uma lente e sensor de imagem de uma câmera de vídeo ou foto. Pode-se utilizar esses parâmetros para corrigir a distorção da lente, medir o tamanho de um objeto ou determinar a localização da câmera na cena (Figura 1). Esses métodos são usados em aplicativos como visão de máquina para detectar e medir objetos, robótica, sistemas de navegação e reconstrução de cenas tridimensionais [Mathworks 2018].

Os parâmetros da câmera incluem coeficientes intrínsecos, extrínsecos e de distorção. Para estimar esses parâmetros, é necessários de pontos 3D e 2D de imagens correspondentes. Tais correspondências podem ser obtidas utilizando diversas imagens de um padrão de calibração, como um tabuleiro de damas. Com as correspondências, é possível resolver os parâmetros da câmera. Depois de calibrar uma câmera, para avaliar a precisão dos parâmetros estimados, é possível plotar as localizações relativas da câmera e o padrão de calibração, calcular os erros de reprojeção e os erros de estimativa de parâmetros.[Mathworks 2018]:

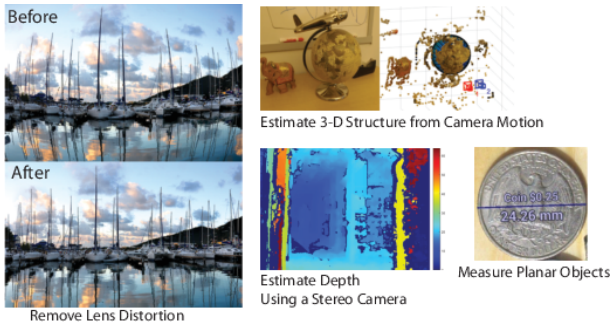

Figure 1: Aplicações da calibração de câmera geométrica.

O consenso de amostra aleatória (RANSAC), é um método iterativo para estimar um modelo matemático de um conjunto de dados que contém *outliers*. O algoritmo RANSAC funciona identificando os outliers em um conjunto de dados e estimando o modelo desejado usando dados que não contêm *outliers*[Mathworks 2018].

Primeiramente o código do RANSAC seleciona aleatoriamente um subconjunto do conjunto de dados; depois ajusta um modelo ao subconjunto selecionado; determina o número de *outliers*; e por fim, repete os passos anteriores para um número determinado de iterações [Mathworks 2018].

Na visão computacional, o RANSAC é usado como uma abordagem robusta para estimar a matriz fundamental em visão estéreo, para encontrar a semelhança entre dois conjuntos de pontos para detecção de objetos baseada em recursos e registrar quadros de vídeo sequenciais para estabilização de vídeo [Mathworks 2018].

O objetivo deste trabalho é apresentar a câmera e a geometria da cena, estimando a matriz de projeção da câmera, que mapeia coordenadas do mundo 3D para as coordenadas da imagem, bem como a matriz fundamental, que relaciona pontos em uma cena a linhas epipolares em outra. A matriz de projeção da câmera e a matriz fundamental podem ser estimadas usando correspondências pontuais. Para determinar a matriz fundamental, a entrada corresponde a 2D pontos em duas imagens. A matriz fundamental pode ser estimada usando as correspondências pontuais de RANSAC [Tompkin 2017].

Interesting Implementation Detail

Este trabalho possui três partes:

1. Estimar a matriz de projeção;
2. Estimar a matriz fundamental;
3. Estimar a matriz fundamental de forma confiável com o RANSAC a partir de correspondências não confiáveis do SIFT.

As partes 1 e 2 já foram disponibilizada prontas, sendo necessário apenas criar a parte 3, que foi implementada no código `ransac_fundamental_matrix.m`. Basicamente, o algoritmo deve estimar uma matriz fundamental baseada na escolha de 8 pontos aleatórios de um conjunto de pontos pertencentes a duas imagens. Essa matriz é utilizada para definir *inliers* que melhoram a precisão de correspondências obtidas pelo algoritmo do SIFT.

Inicialmente, foi definido o número de interações em que o algoritmo deve ser executado, que depende de um número de pontos (linha 1), um limite de erro (linha 2), uma taxa de confiança (linha 3) e uma função utilizando log (linha 4), como descrito a seguir:

```
1 pontPorInter = 8; %quantidade de pontos
2 distThresold = 0.05; %limite aceitavel de erro
3 p = 0.99; % taxa confianca
4 N = log(1-p)/log(1-(1-distThresold).^pontPorInter);
```

Dentro do laço de repetição, por meio da função `randi`, foi definido os 8 pontos aleatórios de cada conjunto de pontos das imagens (*matches_a* e *matches_b*) (linhas 4, 5 e 6). Com esses pontos foi possível obter a matriz fundamental utilizando uma função denominada `estimate_fundamental_matrix`(linha 9), como demonstra o código a seguir:

```
1 (...)
2 for i=1: N
3   % Escolher aleatoriamente 8 pontos de cada matches
4   ponts = randi(size(matches_a,1), [pontPorInter,1]);
5   ponts_match_a = matches_a(ponts,:);
```

```
6   ponts_match_b = matches_b(ponts,:);
7
8   %Obter a matriz fundamental
9   Fmatrix = estimate_fundamental_matrix(ponts_match_a,
        ponts_match_b)
10  (...)
```

Sabendo a matriz fundamental, foi necessário definir uma métrica capaz de encontrar os *inliers* do conjunto de pontos das imagens (*matches_a* e *matches_b*). Essa métrica, consistiu em um somatório que multiplica `xb`, matriz fundamental e `xa'`, onde `xb` e `xa` são, respectivamente, os *matches* b e a com uma dimensão a mais contendo apenas o valor 1. As linhas 4, 5, 7, 8, 10 e 11 do código a seguir demonstra como foi obtido as matrizes `xa` e `xb`. Já a linha 15 demonstra a métrica utilizada:

```
1   (...)
2
3   % defirnir matriz auxiliar para utilizar na metrica de
        distancia
4   tamMatches_a = size(matches_a, 1); %tamanho da matriz
        mataches_a
5   tamMatches_b = size(matches_b,1); % tamanho da matriz
        matches_b
6   %criar um vetor de 1 com tamanho dos matches_a e b
7   one_matches_a = ones(tamMatches_a,1);
8   one_matches_b = ones(tamMatches_b,1);
9   %criar matriz igual a matches_a e b com uma coluna a mais
        contendo valores 1
10  xa = [matches_a one_matches_a];
11  xb = [matches_b one_matches_b];
12
13  (...)
14  %calcular metrica de distancia baseada na matriz
        fundamental para contar o numero de inliers (xb * F *
        xa')
15  metrica = sum((xb .* (Fmatrix * xa')'),2); % xb * F * xa'
16
17  (...)
```

Com a métrica, foi possível determinar o vetor de *inliers* considerando um limite de erro pré-definido, e a sua quantidade. A linha 3 a seguir demonstra como foi determinado esse vetor, e a linha 5 como foi definido o tamanho dos *inliers*:

```
1   (...)
2    %vetor de inliers
3     limEr = find(abs(metrica) <= distThresold);
4     % encontra o numero de inliers
5     inliers = size( limEr , 1);
```

```
6   (...)
```

A cada interação do algoritmo, foi armazenada a matriz fundamental e o vetor que apresentou o maior número de *inliers*. Para isso, foi feito uma comparação simples, como demonstra a condição das linhas 2 a 6 a seguir:

```
1 (...)
2  if (inliers > maxInliers)
3      Best_Fmatrix = Fmatrix;
4      maxInliers = inliers
5      %armazenar o vetor de inliers correpondete ao
           maxInliers
6      currentInliers  = limEr;
7   end
8   (...)
```

Ao final, foi retornado na função os *inliers* encontrados correspondentes aos *matches* a e b. Para isso, foi armazenado os *matches* nas posições dos *inliers* encontrados, e a quantidade desses *matches* equivale ao número máximo de *inliers* encontrado, como mostra a linha 3 e 4 a seguir:

```
1 (...)
2 %encontrar os inliers a e b de acordo com o maximo
     inliers encontrado
3 inliers_a = matches_a(currentInliers(1:maxInliers),:);
4 inliers_b = matches_b(currentInliers(1:maxInliers),:);
5 (...)
```

A Result

O primeiro resultado foi satisfatório, pois as correspondências obtidas utilizando o RANSAC implementado foram melhores do que utilizando apenas o SIFT. A Figura 2 mostra o resultado da imagem Mount Rushmore utilizando apenas o SIFIT. Já a Figura 3 apresenta a mesma imagem, porém utilizando o RANSAC.

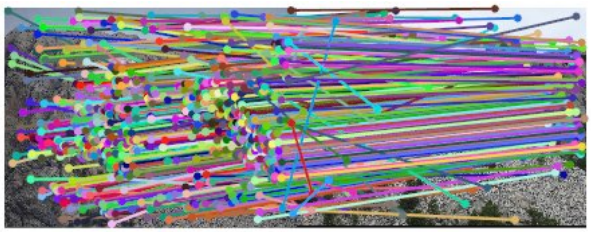

Figure 2: Correspondências Mount Rushmore utilizando SIFIT.

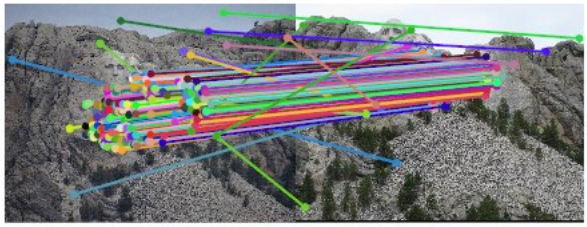

Figure 3: Correspondências Mount Rushmore utilizando RANSAC implementado.

É possível observar que o uso do RANSAC melhora várias correspondências erradas do SIFIT, entretanto piora outras. Para aprimorar esse resultado, foi re-estimada a matriz fundamental usando todos os *inliers* encontrados pelo RANSAC. A Figura 4 mostra o resultado obtido.

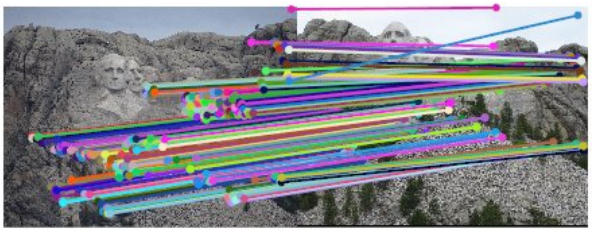

Figure 4: Correspondências Mount Rushmore obtida pela re-estimação da matriz fundamental usando todos os inliers.

É possível observar que os resultados após a re-estimação da matriz fundamental foi extremamente satisfatório, pois quase todas as correspondências foram estimadas de forma correta.

References

[Mathworks 2018]MATHWORKS. *Using RANSAC for estimating geometric transforms in computer vision*. 2018. https://www.mathworks.com/discovery/ransac.html. Acessado em 02-06-2018.

[Mathworks 2018]MATHWORKS. *What Is Camera Calibration?* 2018. https://www.mathworks.com/help/vision/ug/camera-calibration.html. Acessado em 02-06-2018.

[Tompkin 2017]TOMPKIN, J. *Camera Calibration and Fundamental Matrix Estimation with RANSAC*. 2017. http://cs.brown.edu/courses/cs143/proj5a/. Acessado em 02-06-2018.