Project 4 Writeup

Instructions

- Describe any interesting decisions you made to write your algorithm.
- Show and discuss the results of your algorithm.
- Feel free to include code snippets, images, and equations.
- Use as many pages as you need, but err on the short side If you feel you only need to write a short amount to meet the brief, th
- **Please make this document anonymous.**

Nomes

- Caio Cesar Hideo Nakai
- Gabriel Choptian

Introdução

O objetivo deste trabalho é estimar a matriz fundamental que relaciona pontos em uma cena a linhas epipolares em outra. A matriz fundamental pode ser estimada usando pontos correspondentes. Para estimar a matriz fundamental, a entrada corresponde a pontos 2d em duas imagens. A matriz fundamental foi estimada utilizando os pontos correspondetes do SIFT juntamente com o RANSAC.

Para encontrar uma boa matriz fundamental com a maior quantidade de *inliers* utiliza-se o algoritmo RANSAC. Dessa forma, é possível filtrar os *matches* errados e alcançar uma quase perfeita correspondência ponto a ponto.

O RANSAC é um método iterativo para estimar os parâmetros de um modelo matemático, ele é considerado um algoritmo robusto para lidar com *outliers*, pois escolhe o melhor modelo com a maior quantidade de *inliers* baseado em um *threshold*, no entanto, o tempo computacional cresce rapidamente dependendo da quantidade de *outliers* e o número de parâmetros. As aplicações mais comuns de utilizá-lo é estimar a matriz fundamental e computar homografia. O algoritmo é simples e pode ser definido em três passos, que são repetidos até encontrar o melhor modelo com a maior confiança:

1. Sample (randomly) the number of points s required to fit the model
2. Solve for model parameters using samples
3. Score by the fraction of inliers within a preset threshold of the model

Detalhes Interessantes de Implementação

A quantidade de iterações do RANSAC para estimar uma boa matriz fundamental pode ser determinada através da fórmula:

$$N = \frac{log(1-p)}{log(1-(1-e)^s)}$$

No entanto, a taxa de *outlier* precisa ser conhecida, como não conhecíamos a taxa de *outlier* de todas as imagens foi utilizada uma quantidade fixa de iterações (1.000 e 10.000), porém para a imagem do Mount Rushmore com a taxa de *outlier* conhecida em 30% o algoritmo apresentou bons resultados com a fórmula.

O erro foi calculado através multiplicação entre o *matche* de uma imagem com a coordenada homogênea, a matriz fundamental e a tranposta do *matche* da outra imagem também com a coordenada homogênea, representado no código abaixo pelas linhas 13, 15 e 17. O limite ou *threshold* foi determinado de forma empírica, ou seja, foram testados diferentes valores até encontrar um resultado satisfatório.

Para armazenar os matches que possuíam um erro menor que o *threshold* definido, foi criado um *array* de zeros com o tamanho da quantidade de matches, assim nas posições onde o erro era menor do que o *threshold* foi colocado o valor 1, dessa forma, para recuperar os valores dos inliers foi utilizada a função *find* que devolve os indíces das posições do *array* onde há o valor 1, mostrado nas linhas 19, 25 e 27 do código abaixo.

```
1     for i = 1:1000
2       random_points = randperm(max_points_size,
           sampled_points);
3
4       points_random_a  = matches_a(random_points, :);
5
6       points_random_b  = matches_b(random_points, :);
7
8       Temp_Fmatrix = estimate_fundamental_matrix(
           points_random_a, points_random_b);
9
10      inliers = zeros(max_points_size,1);
11
12      for j = 1:max_points_size
13          matrix_A = [matches_a(j,:), 1]';
14
15          matrix_B = [matches_b(j,:), 1];
16
17          err = matrix_B * Temp_Fmatrix * matrix_A;
18
19          inliers(j) = abs(err) < threshold;
20      end
21
```

```
22        if sum(inliers) > size(inliers_a,1)
23          Best_Fmatrix = Temp_Fmatrix;
24
25          inliers_a = matches_a(find(inliers),:);
26
27          inliers_b = matches_b(find(inliers),:);
28        end
29      end
```

Resultados

As Figuras 1, 2, 3, 4, 5 são os resultados obtidos. Os parâmetros utilizados para obter estes resultados nas Figuras 1, 2, 3 foram:

- Threshold: 0.01
- Nº de iterações: 1000

E para as Figuras 4 e 5 foram:

- Threshold: 0.001
- Nº de iterações: 10000

Figure 1: Episcopal Gaudi

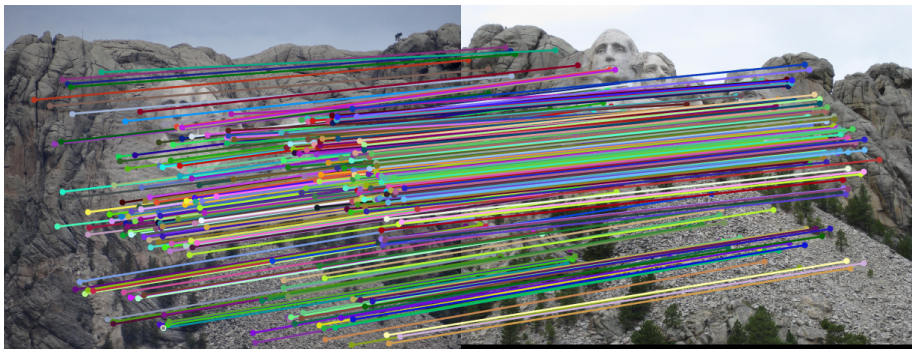

Figure 2: Mount Rushmore

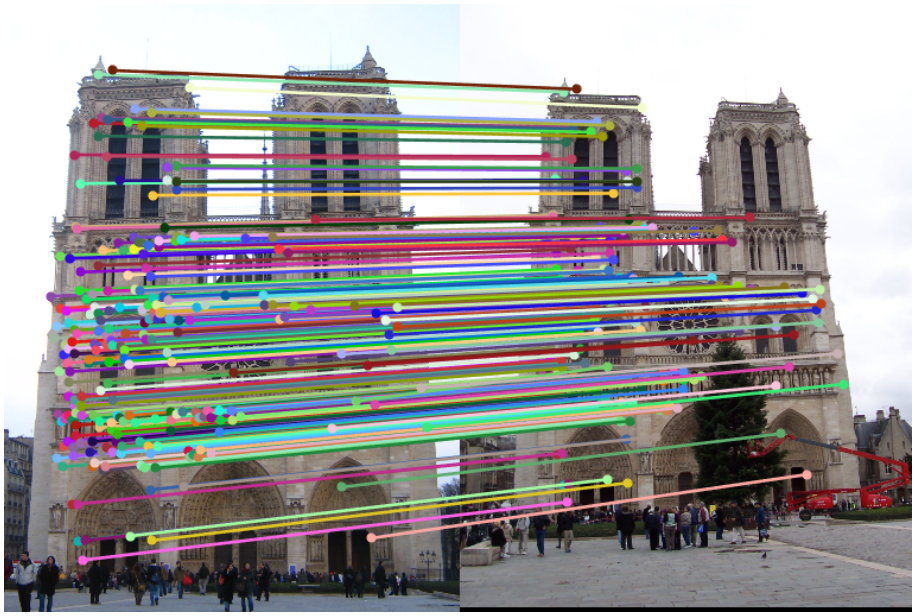

Figure 3: Notre Dame

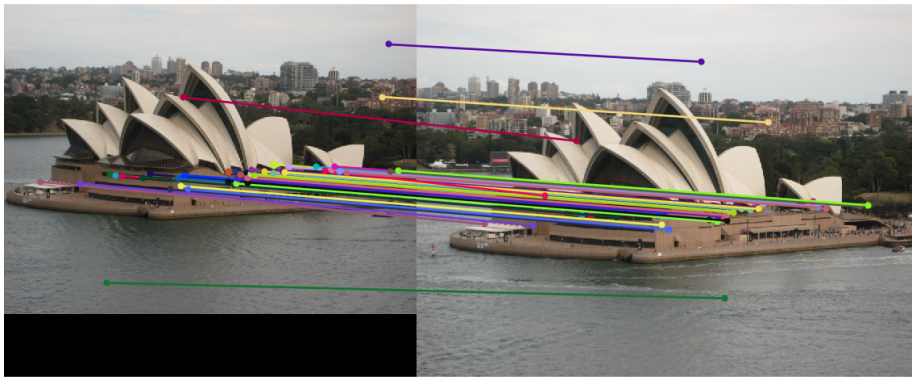

Figure 4: Opera House

Figure 5: Woodruff Dorm