

## Project 5 Image Mosaicing using RANSAC to Robustly Estimate Homographies

### Objetivo

O objetivo deste trabalho é desenvolver um algoritmo que receba um conjunto de imagens, para gerar um mosaico (panorâmica) de imagens semelhantes. Na qual, este mosaico é composto por combinações de várias imagens semelhantes, entretanto em com pontos focais distintos. Após a execução do código, é gerado uma única imagem maior e completa, contendo a melhor combinação das imagens de uma mesma cena. Para isto é necessário estimar uma matriz de homografia que relaciona os pontos das imagens entre os planos, por meio das correspondências geradas pelas funções *SIFT* e *RANSAC*.

### O Algoritmo RANSAC

O algoritmo RANSAC foi introduzido pela primeira vez por Fischler e Bolles em 1981 como um método para estimar os parâmetros de um determinado modelo, a partir de um conjunto de dados contaminados por grandes quantidades de outliers (valores muito diferentes do valor médio).

É um algoritmo iterativo, não determinístico, que usa least-squares para estimar os parâmetros do modelo. A premissa básica do RANSAC é a presença no conjunto de dados de ambas as observações que se ajustam ao modelo (inliers) e aquelas que diferem dos valores (outliers). As fontes de dados que não se encaixam no modelo são erros grosseiros (erros de medição), ruído ou outras perturbações. Os dados de entrada do algoritmo são: um conjunto de dados e um modelo matemático que será correspondido ao conjunto de dados. A vantagem deste método é que a porcentagem de outliers que podem ser entregues pelo RANSAC pode ser maior que 50 por cento de todo o conjunto de dados (MURRAY TORR, 1997).

### Detalhes da Implementação

Toda a implementação realizada para este projeto, consistiu em concluir o trecho de código no arquivo *ransac\_est\_homography.m*, presente no seguinte diretório *Image\_Mosaicing/code*. Para isso foi proposto uma sequência de passos à serem seguidos, que são:

- Passo 1: Selecionar quatro pares de características aleatoriamente;
- Passo 2: Computar a homografia;
- Passo 3: Computar os inliers;

- Passo 4: Repetir os passos 1,2 e 3;
- Passo 5: Manter o maior conjunto de inliers;
- Passo 6: Recalcular a estimativa H dos mínimos quadrados em todos os inliers.

Os passos descritos anteriormente podem ser observado na forma de algoritmo através do trecho de código a seguir.

```

1 #Inicializacao das variaveis
2 N = size(y1, 1); #Variavel para gerar pontos aleatorios
3 iteracoes = 1000; #Numero de iteracoes
4 score = zeros(iteracoes, 1); #Vetor para guardar os
   scores
5 possiveisPontos = cell(iteracoes, 1); #Armazenar a a
   diferenca das distancias
6 homografias = cell(iteracoes, 1);      #Armazenar as
   homografias estimadas
7
8 #Init iteration
9 for t = 1 : iteracoes
10    # Selecionar aleatoriamente os pontos
11    subsets = randperm(N, 4);
12
13    #Estimar Homografia
14    homografias{t} = est_homography(x1(subsets),y1(
       subsets),x2(subsets),y2(subsets));
15
16    #Aplicando a Homografia
17    [x1Est, y1Est] = apply_homography(homografias{t}, x2,
       y2);
18
19    #calcula a diferenca das distancias ao quadrado, se
      for menor que o thresh, armazena no vetor inliers
20    possiveisPontos{t} = ((x1 - x1Est).^2 + (y1 - y1Est)
      .^2) <= thresh.^2;
21
22    #calcula a soma de quantidade de matches no inlier
23    score(t) = sum(possiveisPontos{t}) ;
24 end

```

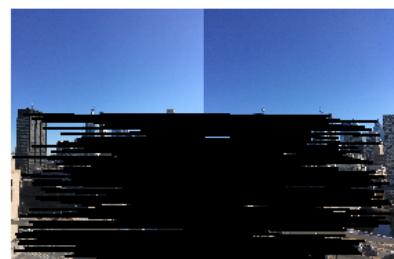
## Resultados

Para a primeira execução foram submetidas as imagens do conjunto *./Data/balcony*, após o processamento, é gerado a imagem final por meio das combinações das imagens de entrada. A seguir, são apresentados os resultados da 1<sup>a</sup> iteração, pela Figura 1.

**280 Original matches**



**209 (74.64%) inliner matches out of 280**



**Mosaiced Image in iteration 1**



Figure 1: Resultados da primeira iteração (Balcony).

Para a segunda iteração ainda sobre as imagens do conjunto *balcony*, são apresentados pela Figura 2.

**307 Original matches**



**228 (74.27%) inliner matches out of 307**



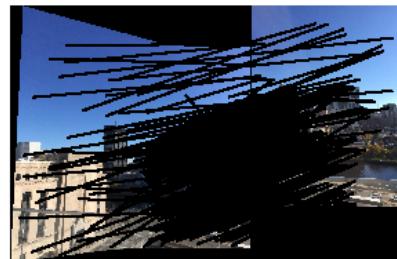
**Mosaiced Image in iteration 2**



Figure 2: Resultados da segunda iteração (Balcony).

Na quarta iteração, foram obtidos os seguintes resultados, conforme apresentado pela Figura 3.

**296 Original matches**



**211 (71.28%) inliner matches out of 296**



**Mosaiced Image in iteration 4**



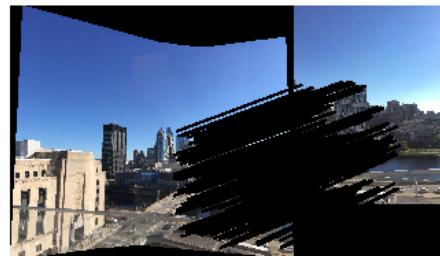
Figure 3: Resultados da quarta iteração (Balcony).

Para a quinta e última iteração, os resultados obtidos podem ser visualizados a seguir pela Figura 4.

**336 Original matches**



**194 (57.74%) inliner matches out of 336**



Mosaiced Image in iteration 5

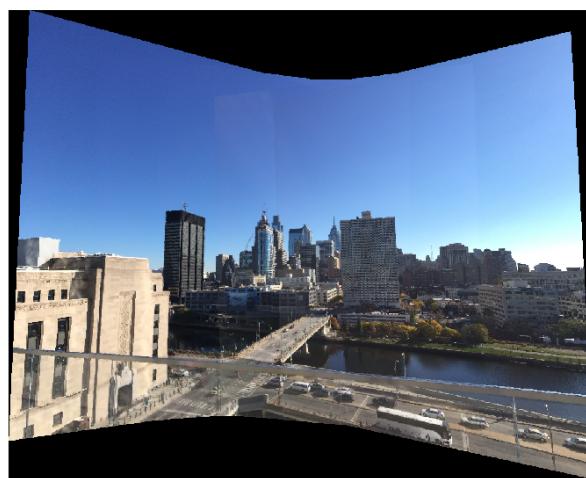


Figure 4: Resultados da quinta e última iteração (Balcony).

Selecionando outro conjunto de entrada para a execução do o algoritmo, como sendo as imagens presentes no diretório *./Data/Re-Estrada*, sendo estas 3 imagens de autoria própria, apresentadas pela Figura 5.



Figure 5: Imagem presentes no conjunto Re-Estradas.

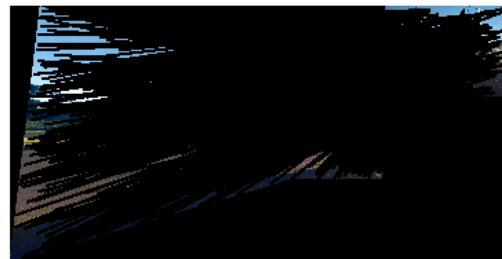
Os resultados obtidos da primeira iteração para o conjunto *Re-Estradas*, são apresentados pela Figura 6.



Figure 6: Resultados da primeira iteração (Re-Estradas).

Para a terceira e última iteração, os resultados finais são apresentados pela Figura 7.

**762 Original matches**



**125 (16.40%) inliner matches out of 762**



**Mosaiced Image in iteration 3**



Figure 7: Resultados da terceira iteração (Re-Estradas).

Para o último conjunto de imagens utilizadas como entrada do algoritmo, foram selecionadas as imagens do diretório *./Data/Re-Sala*. Em especial, para este conjunto, era esperado que o algoritmo tenha problemas para combinar as imagens, uma vez que as imagens foram realizadas em um ambiente com muitos objetos semelhantes (computadores) próximos à câmera. As imagens selecionadas para esta execução podem ser observadas a seguir pela Figura 8.

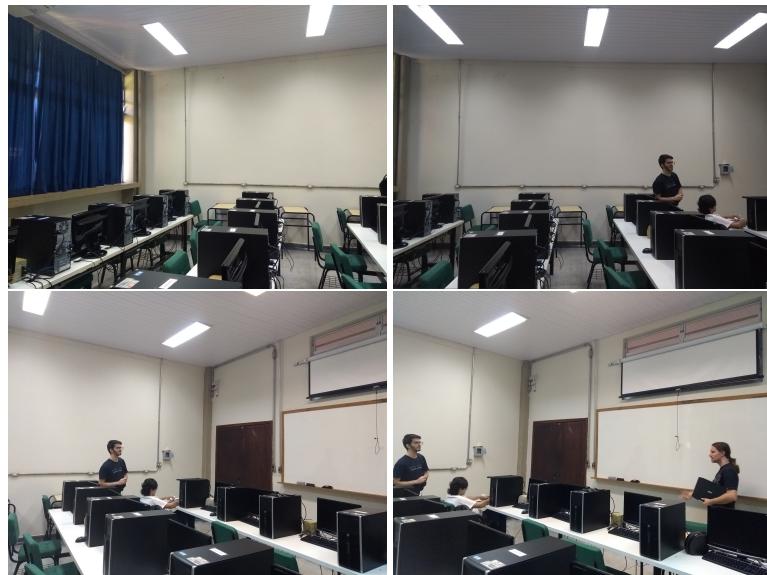
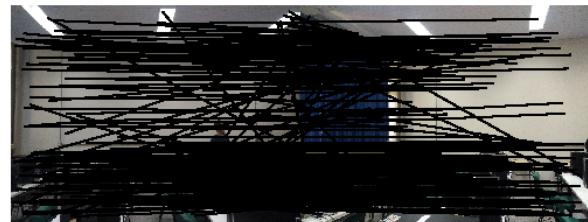


Figure 8: Imagens do conjunto Re-Salas.

A seguir são apresentados os resultados referente à primeira iteração do algoritmo, sendo apresentado pela Figura 9. Os resultados da terceira iteração são apresentados pela Figura 7.

Como observado na primeira iteração, é notável que o algoritmo obteve um resultado final satisfatório em combinar as imagens mesmo obtendo 43% de pontos inliers. Entretanto, na terceira iteração, o algoritmo apresentou problemas em combinar as imagens, obtendo um match de apenas 3% de correspondência. O resultado final pode ser observado pela Figura 11.

**243 Original matches**



**104 (42.80%) inliner matches out of 243**



**Mosaiced Image in iteration 1**

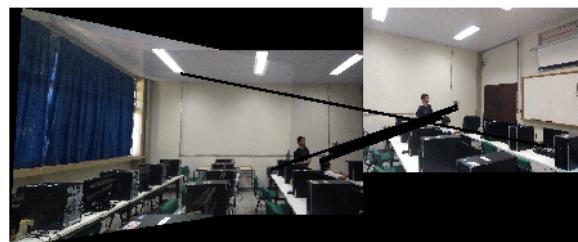


Figure 9: Resultados da primeira iteração (Re-Salas).

**283 Original matches**



**9 (3.18%) inliner matches out of 283**



**Mosaiced Image in iteration 3**

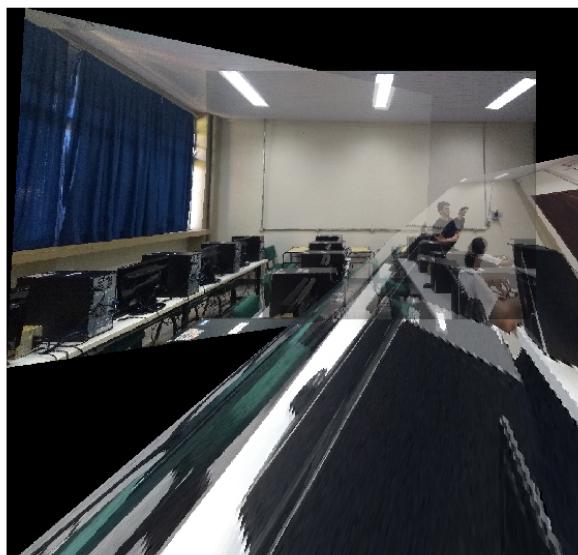
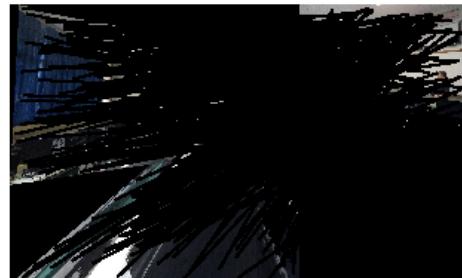
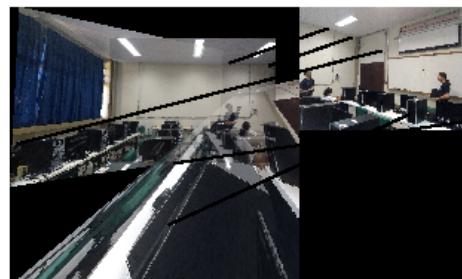


Figure 10: Resultados da terceira iteração (Re-Salas).

**402 Original matches**



**6 (1.49%) inliner matches out of 402**



**Mosaiced Image in iteration 4**

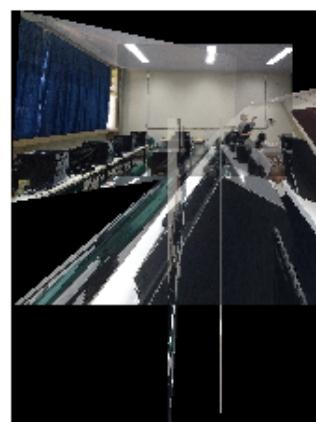


Figure 11: Resultados da quarta iteração (Re-Salas).