

## Project 4 Fundamental Matrix Estimation with RANSAC

### Objetivos

O objetivo é criar um algoritmo que recebe um conjunto de pontos relacionados de duas imagens e que descarte os pontos do conjunto que não são confiáveis, ou seja, os pontos que podem estar relacionados erroneamente. Para isso devemos estimar a matriz fundamental, que relaciona pontos em uma imagem a linhas epipolares em outra.

### 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).

O algoritmo RANSAC é essencialmente composto por duas etapas que são repetidas em um processo iterativo:

- Hipótese
- Teste

A informação apriori, que é usada no processo de adaptação do modelo, inclui:

1. Número mínimo de pontos necessários para ajustar o modelo;
2. Número mínimo de iterações;
3. Parâmetro (threshold) que determina o limiar que divide os inliers de outliers no processo de teste de modelo hipotético;

Estas informações são ajustáveis e o resultado obtido tende a variar muito de uma imagem para outra mesmo com os valores iguais.

## Detalhes de Implementação

Os 3 valores citados anteriormente podem ser alterados em nosso código. Eles foram adicionados no arquivo ransacFundamentalMatrix.m que foi disponibilizado em um projeto base para construção do algoritmo ("projCameraGeom.zip").

```

1 num_rand_pts = 8;
2 ransac_iter = 2000;
3 inlier_threshold = 0.08;
```

Conforme citado anteriormente, o algoritmo RANSAC pode ser basicamente dividido em duas etapas. A fase da hipótese está relacionado ao conceito do conjunto mínimo de amostras. O primeiro conjunto mínimo de amostras é selecionado aleatoriamente a partir do conjunto de dados de entrada e os parâmetros do modelo são calculados usando apenas os elementos do conjunto. Assim, a primeira fase começa com a seleção de um número necessário e mínimo de observações do conjunto de dados. Com base nessas observações selecionadas, o modelo de saída (hipotético) é estimado. Todos os dados restantes são testados em termos de adequação ao modelo hipotético. Se um ou ambos os pontos selecionados estiverem sobrecarregados com outliers, então o modelo hipotético não se ajustará ao resto dos dados. Portanto, o algoritmo irá pular este modelo e escolher aleatoriamente outro par de pontos para outro modelo hipotético. Segue abaixo o trecho do código responsável por esta etapa:

```

1 for x=1:1:ransac_iter
2     rand_int_pts = randi(sa, num_rand_pts, 1);
3     rand_a = matches_a(rand_int_pts, :);
4     rand_b = matches_b(rand_int_pts, :);
5     curr_F = estimate_fundamental_matrix(rand_a, rand_b);
```

Na etapa de teste, o RANSAC verifica iterativamente quais observações de todo o conjunto de dados são consistentes com o modelo hipotético. Isso requer determinar o valor do parâmetro threshold especificando a distância máxima de um ponto de teste para um modelo hipotético. Se ele se encaixa no critério de threshold, o ponto é tratado como hipotético. O modelo estimado está correto se tiver um número suficiente de pontos que foram classificados como observações corretas (inliers). O melhor conjunto de observações selecionado de todo o conjunto de dados é chamado de conjunto de consenso (CS). Abaixo segue o trecho do código que verifica quem são os inliers com um valor de threshold pré-definido.

```

1 num_inliers = 0;
2 for y=1:1:sa
3     p1 = [matches_a(y,:) 1];
4     p2 = [matches_b(y,:) 1];
5     corresp = p2*curr_F*p1';
6     if abs(corresp) < inlier_threshold
7         num_inliers = num_inliers+1;
8     end
```

9      end

Depois é verificado qual das matrizes (f) é a que posse o melhor resultado, ou seja, a que possua o maior número de inliers. (*Best\_Fmatrix*)

```

1 if num_inliers>max_inliers
2     max_inliers = num_inliers
3     Best_Fmatrix = curr_F;
4     inliers_a = [];
5     inliers_b = [];
6     for y=1:1:sa
7         p1 = [matches_a(y,:)]';
8         p2 = [matches_b(y,:)]';
9         corresp = p2*curr_F*p1';
10        if abs(corresp) < inlier_threshold
11            inliers_a = [inliers_a; matches_a(y,:)];
12            inliers_b = [inliers_b; matches_b(y,:)];
13        end
14    end
15 end

```

## Resultados

É possível observar o funcionamento do algoritmo analisando as imagens de entrada e saída, a Figura 1 representa a imagem de entrada, obtida por angulos diferentes.

Após o processamento do algoritmo é exibido as imagens de saída, mostrando os pontos relacionado entre as duas imagens de saída, a saída obtida pode ser observado na Figura 2.



Figure 1: *Left:Image one* Imagens de entrada. *Right: Image two*

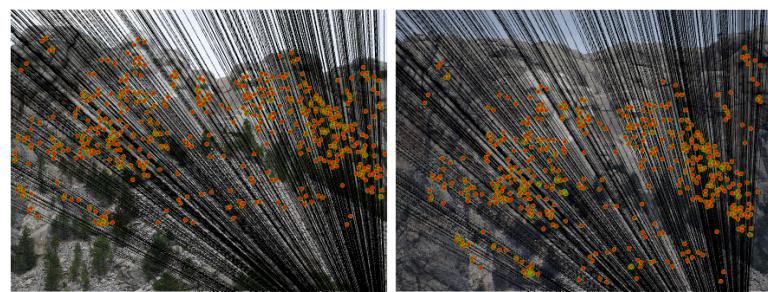


Figure 2: *Left: Image one* Imagens de saída. *Right: Image two*