Trabalho 4 - MC920

Nome: Felipe Pessina RA: 171214

June 10, 2019

1 Introduction

O objetivo deste trabalho é a identificação de pontos de interesse e descritores invariantes locais em 2 imagens, e utilizá-los para criar uma imagem panorâmica a partir das imagens originais. Um dos melhores exemplos é a Foto1A e Foto1B, que temos como resultado uma ótima imagem panorâmica que analisaremos neste relatório. Para construirmos a imagem panorâmica, seguimos as seguintes etapas: transformação em grayscale, aplicação do algoritmo ORB para identificação de pontos de interesse e descritores locais, obtenção dos matchs entre os pontos de interesse das imagens, criação da matriz de homografia e por fim aplicamos a transformação com a matriz de homografia e unimos as imagens

2 Implementação

Neste projeto fizemos uso da linguagem Python, e das seguintes bibliotecas para a leitura, tratamento e escrita das imagens: numpy, opencv.

A primeira etapa do projeto é realizar a leitura das imagens no formato JPG com o comando:

```
img = cv2.imread('nome-imagem', 0)
```

Nesta leitura devemos ler com o parâmetro 0, que indica que nossa leitura será em grayscale. Pois futuramente iremos usar o algoritmo ORB que exige uma imagem em grayscale. A biblioteca do OpenCv tem 3 opções de leitura como descrito abaixo:

<0 loads the image as is (including the alpha channel if present)

0 loads the image as an intensity one

>0 loads the image in the BGR format

Após a leitura da imagem, iremos identificar os pontos de interesse e descritores locais com o algoritmo ORB. Para isto utilizamos os seguinte comando:

```
orb = cv2.ORB_create()
kp = orb.detect(img,None)
kp, des = orb.compute(img, kp)
```

Aqui, kp são os keypoints e des são os descritores, que faremos uso nos nossos próximos algoritmos

Com os keypoints e descritores das nossas 2 imagens de entrada, vamos agora determinar os matches e distâncias (hamming) entre eles. Com o uso do algoritmo BFMatcher, com o seguinte comando:

```
bf = cv2.BFMatcher(cv2.NORM_HAMMING)
matches = bf.knnMatch(des1,des2, k=2)
```

Com o uso do bf.knnMatch, obtemos os k melhores matches entre as duas imagens, no caso k=2. Mas iremos refinar ainda mais nossos matches fazendo uma seleção dos matches através das distâncias, com a seguinte condição

```
for m,n in matches:
if m.distance <0.7*n.distance:
good.append(m)
```

Agora com a obtenção dos matches, iremos definir a matriz de homografía. Que relaciona os pixels da imagem 1 com a imagem 2, permitindo que alinhemos uma imagem com a outra. Apesar do nosso refinamento de matches para selecionar os melhores, ainda possuímos matches ruins. Por isto utilizamos o algoritmo RANSAC (Random Sample Consensus) que nos ajuda a obter um bom resultado, mesmo que hajam matches incorreta. Para isto utilizamos os seguintes comandos:

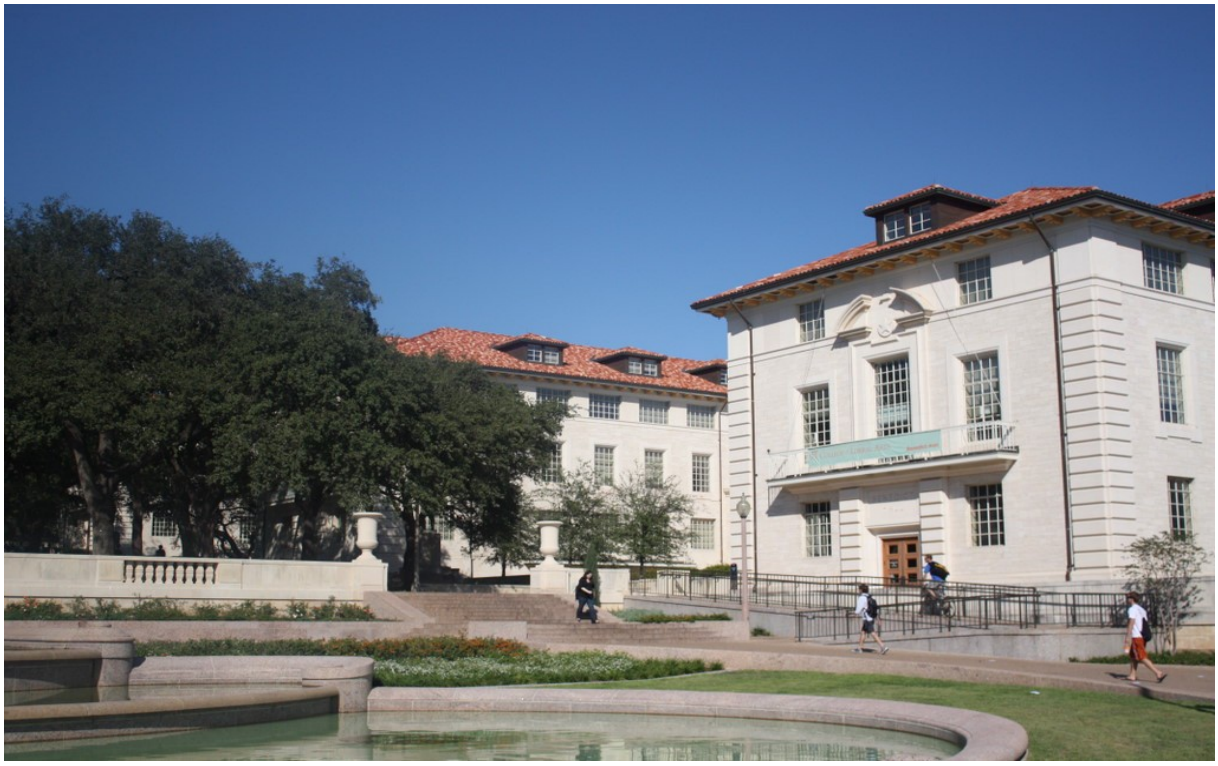
```
src_pts = np.float32([ kp1[m.queryIdx].pt for m in good ]).reshape(-1,1,2)
dst_pts = np.float32([ kp2[m.trainIdx].pt for m in good ]).reshape(-1,1,2)
M, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC,5.0)
```

Agora que possuímos a matriz de homografia, basta aplicarmos a transformação na imagem original e unir elas para obtermos a imagem panorâmica. Realizamos isto com os seguinte comandos:

```
dst = cv2.warpPerspective(img1, M, (gray1.shape[1] + gray2.shape[1], gray2.shape[0]))
dst[0:gray2.shape[0], 0:gray2.shape[1]] = img2
```

3 Resultados

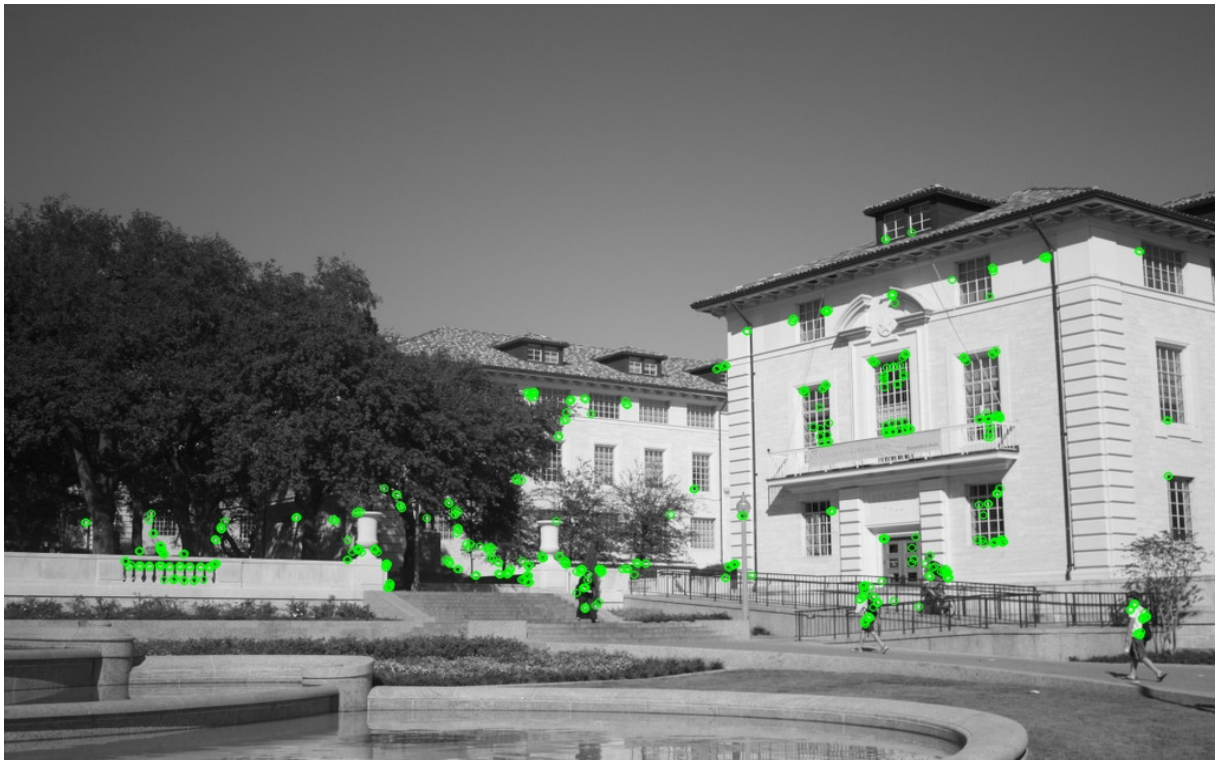
3.1 Imagem Original



Acima vemos as duas imagens que iremos analisar neste relatório, as outras imagens exemplo serão disponibilizadas em uma pasta separada junto com a submissão do relatório. Neste exemplo é fácil ver a composição da imagem panorâmica entre as duas, a primeira imagem fica do lado direito e a segunda do lado esquerdo. Podemos identificar isso através da casa ao fundo das duas imagens, um pouco coberta pelas árvores.

Vamos agora iniciar nossa análise dos resultados, onde o primeiro passo é identificarmos pontos de interesse na imagem e descritores invariantes locais. Basicamente nosso interesse aqui é encontrar pontos da imagem (x, y), que são invariantes a rotação, translação e redimensionamento para permitir que nós criemos imagens panorâmicas mesmo que elas estejam rotacionadas, redimensionadas e transladadas entre si. E não apenas isto mas encontrar os descritores deste ponto, que são features que descrevem estes pontos e nos permite fazer o match entre os pontos das duas imagens

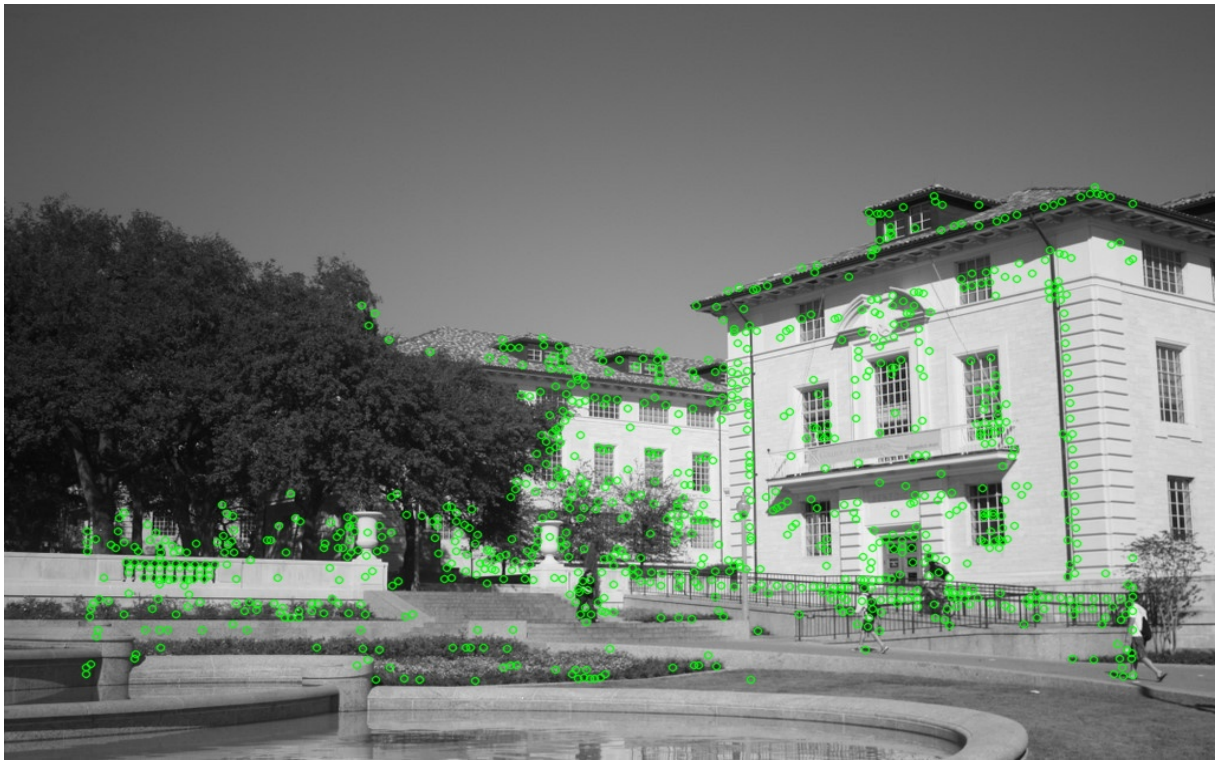
3.2 Detector de Ponto de Interesse ORB



Nosso objetivo aqui é identificar os pontos de interesse nas duas imagens. Estes pontos serão utilizados na próxima etapa para realizarmos os matches entre as imagens, portanto eles devem ser invariantes a rotação, translação e redimensionamento. As regiões que mais satisfazem a essas condições em uma imagem, em geral são regiões de alta frequência na imagem, por isto nas imagens acima vemos que os pontos circulados em verde são em geral bordas de objetos

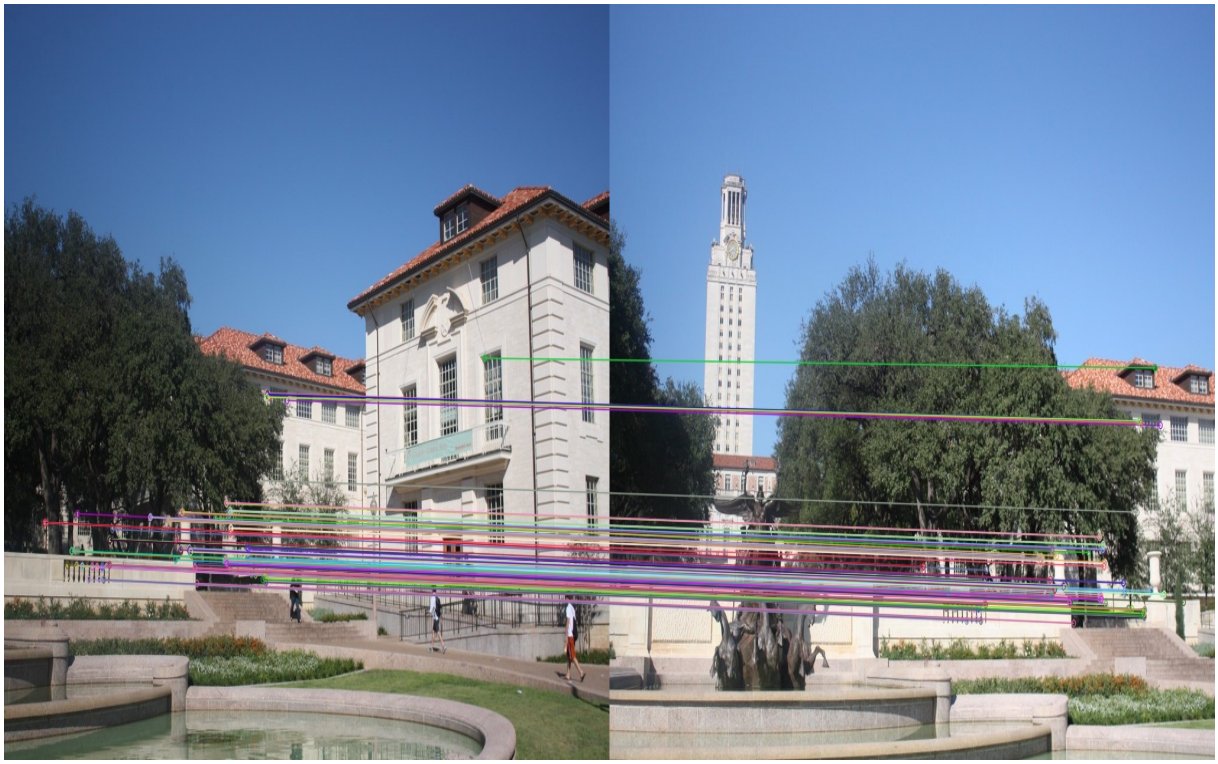
Outra informação interessante sobre o ORB é que ele faz uso dos algoritmos SURF e BRIEF

3.3 BRIEF

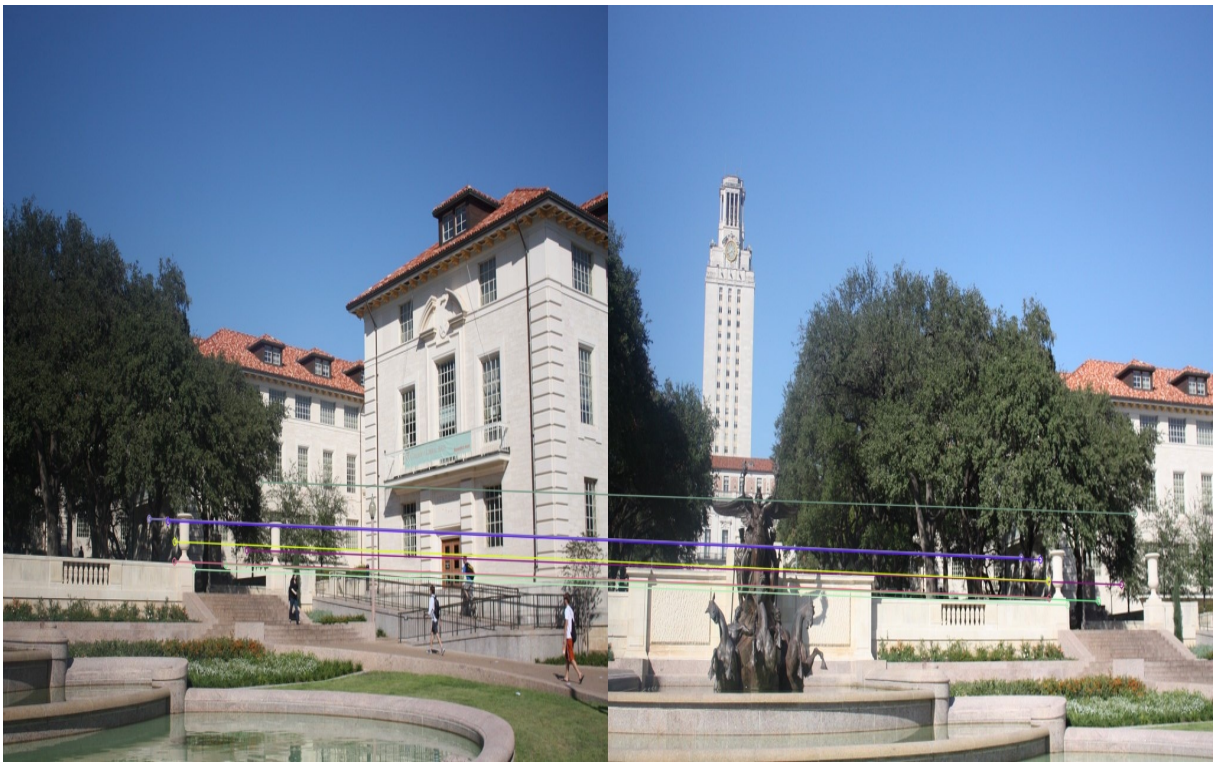


Para propósito de comparação entre os algoritmos, também utilizamos o algoritmo BRIEF, que encontra os descritores dos pontos de interesse. Mas para encontrarmos os keypoints devemos utilizar outro algoritmo, como o ORB. Este algoritmo faz uso da distância de Hamming e por isto tem uma velocidade maior em relação aos outros algoritmos

3.4 BFMatcher



Nesta etapa nós utilizamos o algoritmo BFMatcher, para encontrar os matches entre os pontos de interesse nas duas imagens. Nesta primeira imagem, é difícil identificar quais pontos estão sendo ligados com quais, por isto eu gerei um outra imagem (apenas para visualização) dos melhores matches encontrados entre os pontos filtrados, utilizando a distância de hamming



Agora nesta imagem acima fica mais fácil ver quais pontos foram ligados com quais (matches), este processo é realizado a partir dos pontos de interesse e os descritores, que nos permitem identificar qual o par de cada ponto na outra imagem. Claro que este algoritmo liga pontos que não eram para ser ligados, por isto iremos filtrar os melhores matches e também utilizar o RANSAC para gerar nossa matriz de homografia e minimizar nosso erro

3.5 Matriz de Homografía

Aqui nesta etapa geramos a matriz de homografia. Esta, é uma matriz que relaciona os pixels de uma imagem com os pixels da outra imagem. Sendo o pixel (x1, y1) pertencente a imagem 1, e (x2, y2) um pixel pertencente a imagem 2, a matriz relaciona os pixels da seguinte forma

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = H \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

A matriz de homografia que obtive para esta dupla de imagens foi a seguinte

```
[[ 1.65809137e+00  1.18708630e+00  1.83536370e+02]
 [-4.17795375e-02  2.37083151e+00 -2.72391193e+02]
 [-6.03097669e-06  1.37599384e-03  1.00000000e+00]]
```

Com esta matriz podemos rotacionar os pontos das imagens e alinhá-los. É justamente isto que faremos na próxima etapa.

3.5.1 Alinhamento e criação da imagem panorâmica

Por fim a última etapa basta rotacionarmos as imagens, e sobrepor elas de forma correta utilizando a matriz de homografia e os pontos de interesse. Assim podemos construir uma imagem panorâmica ligando duas imagens de uma mesma região. Como podemos ver acima, há uma linha entre as imagens, que mostra o ponto onde elas foram sobrepostas. E o resultado obtido foi muito bom, as imagens se completam muito bem

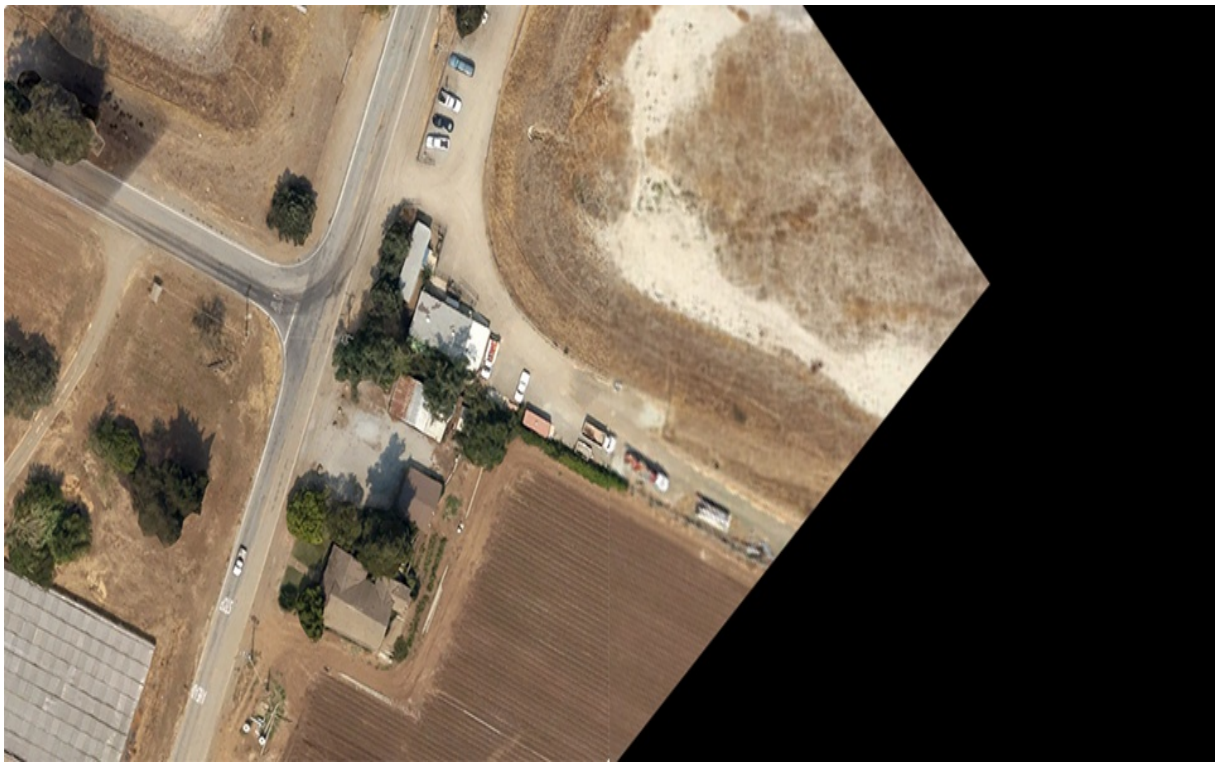
3.6 Conclusão

O objetivo deste projeto era formar uma imagem panorâmica a partir de duas imagens, como podemos ver em nossa última imagem, conseguimos reconstruir uma imagem panorâmica com sucesso a partir de duas imagens

Durante a análise do projeto é interessante destacar o processo do BF-Matcher, que faz uso de descritores invariantes locais e pontos de interesse para realizar os matches. Isto só é possível pois o algoritmos que utilizamos para identificá-los é invariante a rotação, translação e redimensionamento, existem

outros algoritmos em que os pontos de interesse, podem ser perdidos se variarmos alguma destas features.

Como utilizamos um algoritmo como o ORB, podemos também criar uma imagem panorâmica a partir da Foto4, que varia em translação, rotação e também está redimensionada. O resultado final obtido com ela está logo abaixo



References

[1] Identificadores de Ponto de Interesse:
https://docs.opencv.org/3.3.0/db/d27/tutorial_py_table_of_contents_feature2d.html

[2] Feature Matching:
https://docs.opencv.org/3.3.0/db/d27/tutorial_py_table_of_contents_feature2d.html

[3] Matriz de Homografia e Panoramica:
https://docs.opencv.org/3.3.0/d1/de0/tutorial_py_feature_homography.html