Trabalho 5 - Introdução ao Processamento de Imagens Digitais

Nome

Victor Palmerini

RA

178061

Data

17/07/2020

Conteúdos

Introdução

Este é o Trabalho 5 da disciplina ***MC920*** - *Introdução ao Processamento de Imagens Digitais* da **Unicamp** - Universidade Estadual de Campinas.

O principal objetivo deste trabalho é desenvolver uma boa noção sobre registro de imagens, isto é, a partir de 2 imagens obter uma única imagem panorâmica formada pela "junção" dessas imagens.

O trabalho requer por parte do aluno um conhecimento básico da linguagem de programação `python` e bibliotecas que facilitem a manipulação das imagens digitais bem como seus processamentos. No caso deste trabalho, foram usados os pacotes `numpy`, `opencv` e `jupyter notebook`.

Desenvolvimento

Estrutura do Projeto

1. `/notebooks` - é a pasta que contém os notebooks com os algoritmos para serem executados pelo `jupyter notebook`
2. `requirements.txt` - é um arquivo texto que contém as dependências para executar a aplicação (foi gerado pelo pacote `pip` através do comando `pip freeze > requirements.txt`)
3. `Pipfile` e `Pipfile.lock` - são os arquivos relacionados ao `ambiente virtual` aonde as dependências são instaladas e executadas. Isso permite que as dependências sejam instaladas

apenas no ambiente virtual e não no ambiente local.

4. `/images` - pasta com algumas imagens usadas como entrada para execução dos notebooks

Rodando Localmente

Dependências

Para executar os notebooks é necessário instalar as seguintes dependências:

1. `Python 3` - neste projeto foi usada a versão `3.7.7`. Qualquer versão do `python` a partir da `3` é suficiente.
2. `Pipenv` - gerenciador de ambientes virtuais. É equivalente ao `virtualenv`.

Inicialização do Ambiente

Na pasta root do projeto, execute os seguintes comandos em uma `shell`:

1. `pipenv shell` - para iniciar um ambiente virtual localmente
2. `pipenv install -r requirements.txt` - instala no ambiente virtual todas as dependências listadas no arquivo `requirements.txt`

Executando os notebooks

Para executar os notebooks, há 2 caminhos:

1. Executar `jupyter notebook` em uma `shell` para subir um servidor do `jupyterlab` que vai abrir automaticamente o navegador padrão com o ambiente `jupyter` e as pastas e arquivos do projeto.
2. Caso o projeto tenha sido aberto no `Visual Studio Code`, uma opção é instalar a extensão `VS Code Jupyter Notebook`, que permite executar os notebooks no próprio `VS Code`.

Entradas

As entradas dos notebooks serão o `path` das imagens de entrada e o `path` da **pasta** de saída. As imagens de entrada utilizadas estão no formato `.jpg` e podem ser *monocromáticas* ou *coloridas*. Como já foi citado, na pasta `/images` há algumas imagens que podem ser usadas como entrada.

Saídas

As saídas também serão imagens no formato `.png` com a respectiva transformação aplicada.

Implementação

Nesse projeto foi implementada uma técnica apenas de processamento de imagens que foi o registro de 2 imagens em uma só, formando uma imagem panorâmica a partir dessas.

Imagem A

Imagem B

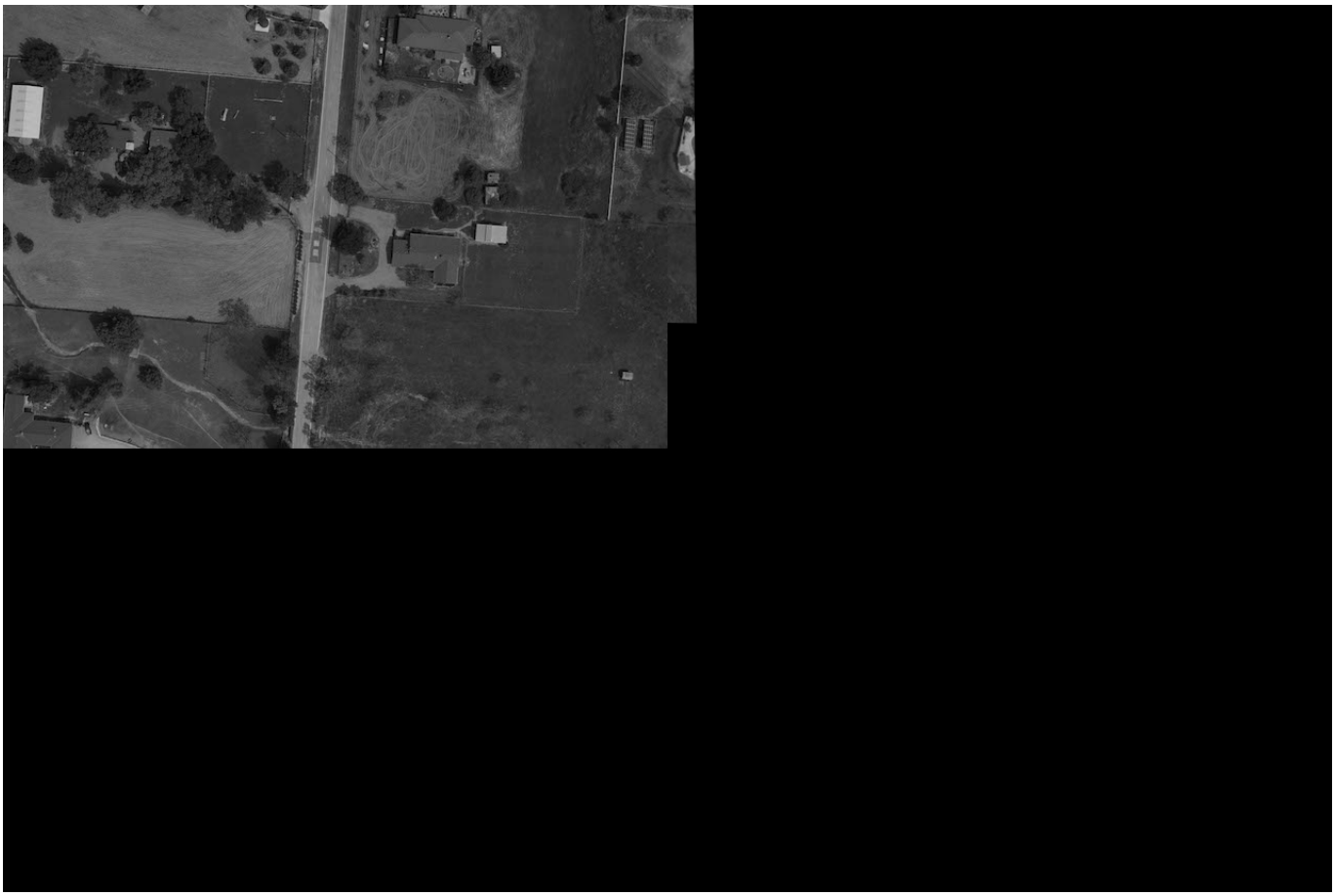

Imagem Panorâmica

Porém, como há diversos algoritmos que podem ser usados no passo de detecção de pontos de interesse e descritores, foi implementado, para cada caso, um notebook utilizando um algoritmo diferente.

Os notebooks que implementam o processamento de formação da imagem panorâmica podem ser divididos nos seguintes passos:

1. Importação das bibliotecas necessárias
2. Importação das imagens de entrada e definição do path de saída
3. Inicialização do objeto relacionado ao algoritmo de detecção de pontos de interesse e descritores (SIFT, SURF e ORB)
4. Aplicação do algoritmo para obtenção desses pontos e descritores
5. Obtenção dos matches dos descritores das 2 imagens (Brute-Force e k-Nearest-Neighbour)
6. Filtro dos matches a partir de um parâmetro de qualidade obtido como entrada do usuário
7. Passo intermediário com a visualização dos matches
8. Obtenção dos pontos de origem e destino para formação da matriz de homografia (utilizando a técnica RANSAC)
9. Aplicação da projeção de perspectiva para alinhamento das imagens
10. Criação da imagem panorâmica

Agora falando um pouco das especifidades de cada algoritmo para detecção dos pontos de interesse e descritores:

SIFT

SIFT (Scale-Invariant Feature Transform) é um algoritmo usado para detectar e descrever features locais em imagens digitais. Como o próprio nome diz, este é um algoritmo que independe da escala e rotação da imagem.

SURF

O SURF (Speeded Up Robust Features) é tido como uma melhoria do SIFT, principalmente em questão de performance. Por ser mais robusto, geralmente é usado em aplicações em tempo real.

ORB

O ORB (Oriented FAST and Rotated BRIEF) é outro algoritmo de detecção de features locais e considerado ainda mais performático que os algoritmos SIFT e SURF.

Obtidos os pontos de interesse e seus descritores pra cada imagem utilizando algum dos algoritmos citados acima, pode-se agora obter os matches entre esses descritores.

Obtenção dos Matches

Para obtenção dos matches entre descritores das 2 imagens, utilizou-se o algoritmo de força-bruta do OpenCV. Este algoritmo basicamente compara cada feature do primeiro conjunto com todas as features do outro conjunto usando algum cálculo de distância. O match retornado para aquela feature é aquele que possui a menor distância.

Para implementar este algoritmo de match, o OpenCV fornece 2 abordagens:

- `cv2.BFMatcher.match()` - retorna todos os matches encontrados
- `cv2.BFMatcher.knnMatch()` - retorna os `k` melhores matches

Após a obtenção dos matches, no caso de não termos feito uma pré-filtragem (utilizando o `knnMatch` por exemplo), podemos ter alguns matches inconsistentes e que podem ser removidos fazendo uma filtragem simples. Basta que consideremos algum parâmetro (geralmente distância) e removemos todos os matches que não atendem àquele parâmetro.

Matriz de Homografia

Para se obter a matriz de homografia, utilizou-se a função `findHomography` do OpenCV. Esta função permite especificar qual o método a ser usado para se obter a transformação de perspectiva. No caso deste projeto, todos os notebooks implementam o método *RANSAC*. Com isso, obtemos a matriz de homografia e podemos aplicar essa transformação para alinhar os planos das imagens.

Alinhamento

Obtida a matriz de homografia, agora basta aplicarmos essa transformação à imagem de saída. O OpenCV nos permite fazer isso a partir da função `warpPerspective`. Ao final, temos a imagem de saída com a devida correspondência e alinhamento das imagens de entrada.

Análise

Podemos dividir a análise em:

- Comparação geral entre os métodos utilizados para obtenção dos pontos de detecção e descritores
- Comparação do método de obtenção dos matches - *brute-force* e _k-nearest e variação dos parâmetros de qualidade (distância máxima e ratio)

Comparação dos métodos de detecção de features

Imagens de entrada:

Imagem A

Imagem B

Executando-se os respectivos notebooks para cada método:

SIFT

```
Key Points A: 4239
Key Points B: 5347
Matches: 2634
```

Imagem Panorâmica - SIFT

SURF

```
Key Points A: 5440
Key Points B: 4766
Matches: 2441
```

Imagem Panorâmica - SURF

ORB

```
Key Points A: 500
Key Points B: 500
Matches: 280
```

Imagem Panorâmica - ORB

Percebe-se que as saídas dos 3 métodos são bem parecidas. Porém, analisando-se as features obtidas, fica claro que o método *ORB* conseguiu ser muito mais eficiente que o *SIFT* e *SURF*. Pois com bem menos pontos de detecção e matches, conseguiu obter um resultado final bem satisfatório. Já os resultados de *SIFT* e *SURF* ficaram bem parecidos assim como as features e matches obtidos.

Comparação dos métodos de obtenção dos matches

Imagens de entrada:


Imagem A


Imagem B

Aqui, foi executado o método *SIFT* utilizando-se filtro a partir da distância dos matches (valor máximo e porcentagem):

Valor Máximo

```
Distância Máxima: 100
Matches: 261
```

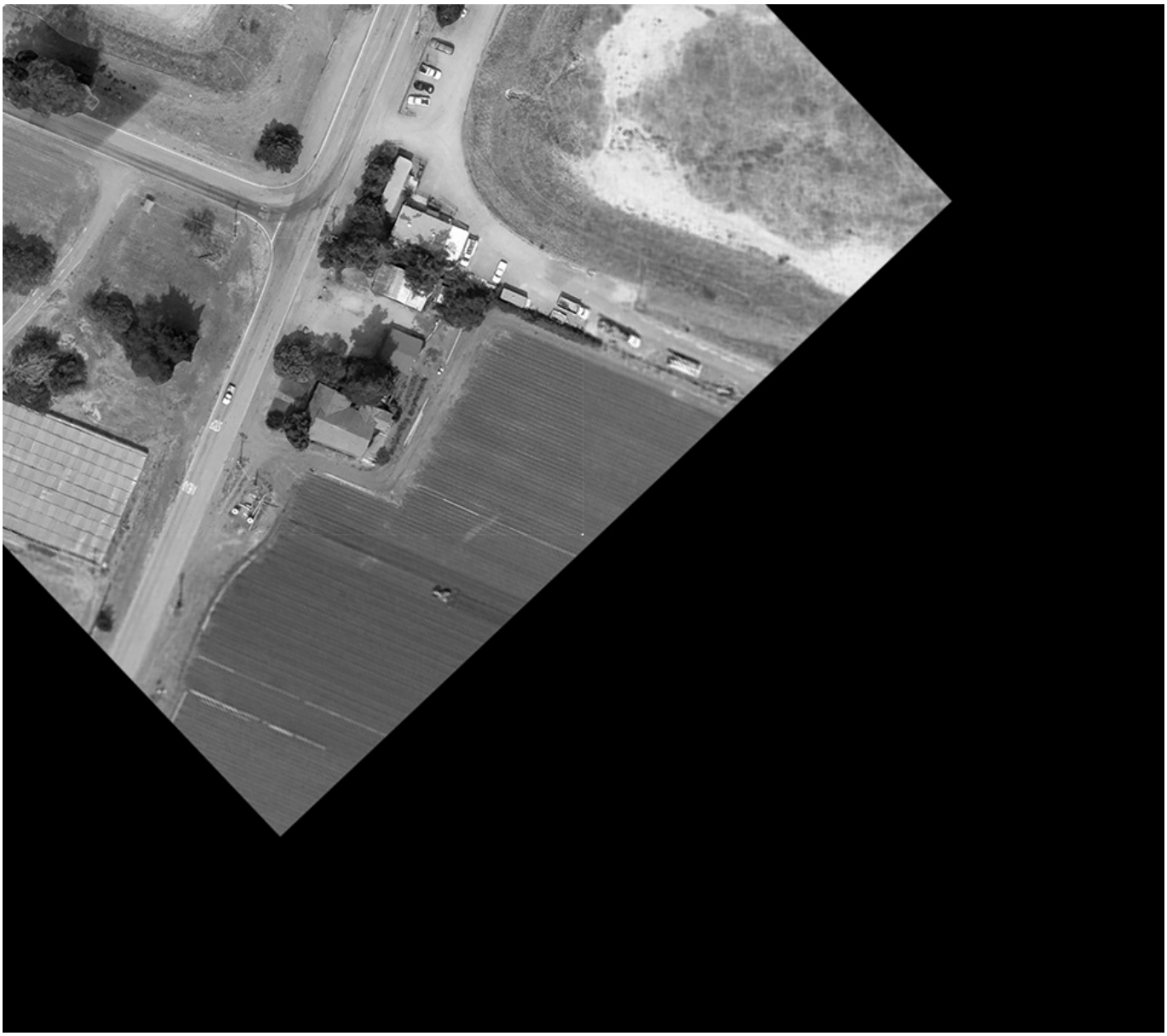

Imagem Panorâmica - SIFT com matches com distância de no máximo 100

```
Distância Máxima: 50
Matches: 87
```

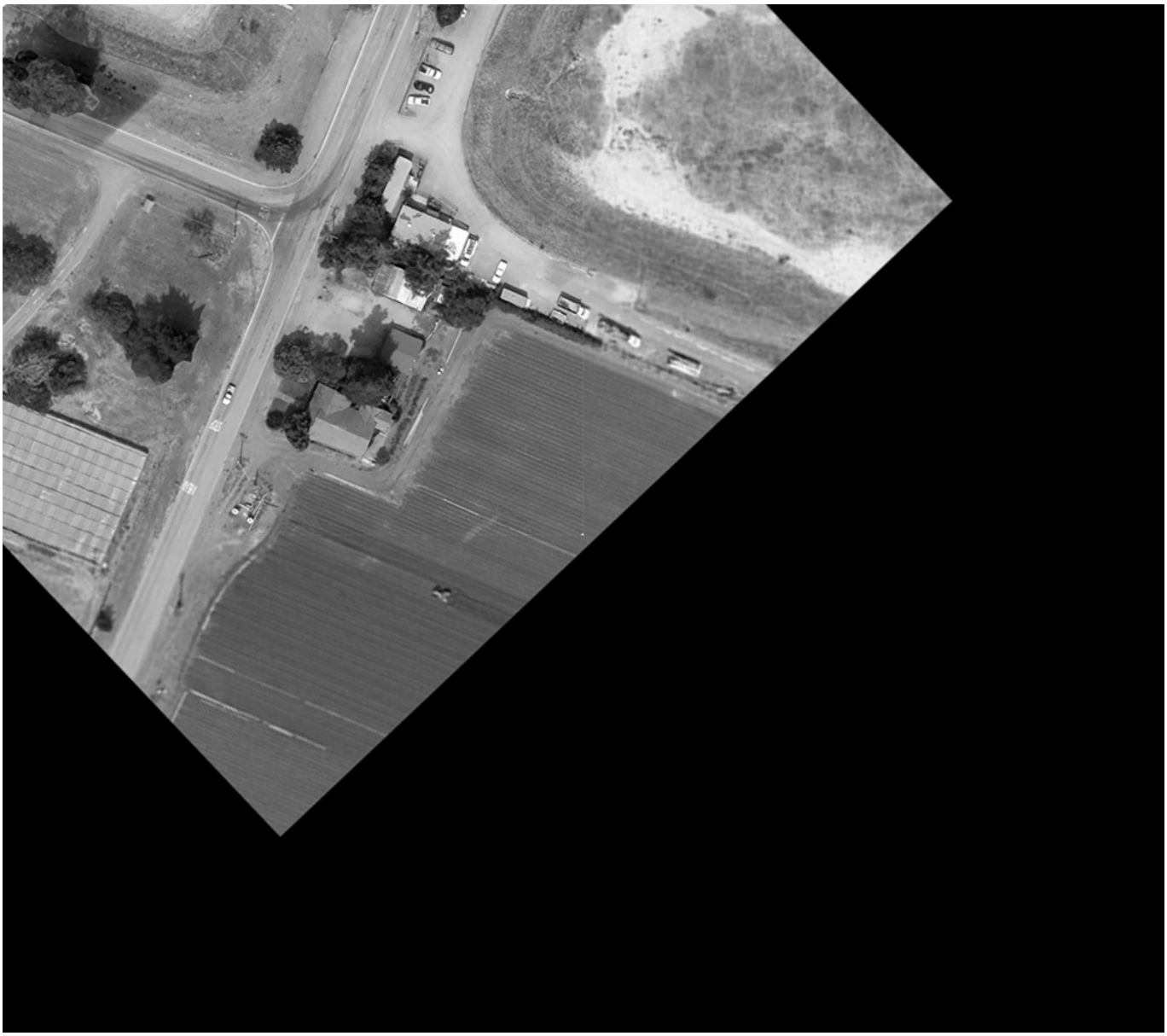

Imagem Panorâmica - SIFT com matches com distância de no máximo 50

```
Distância Máxima: 200
Matches: 362
```

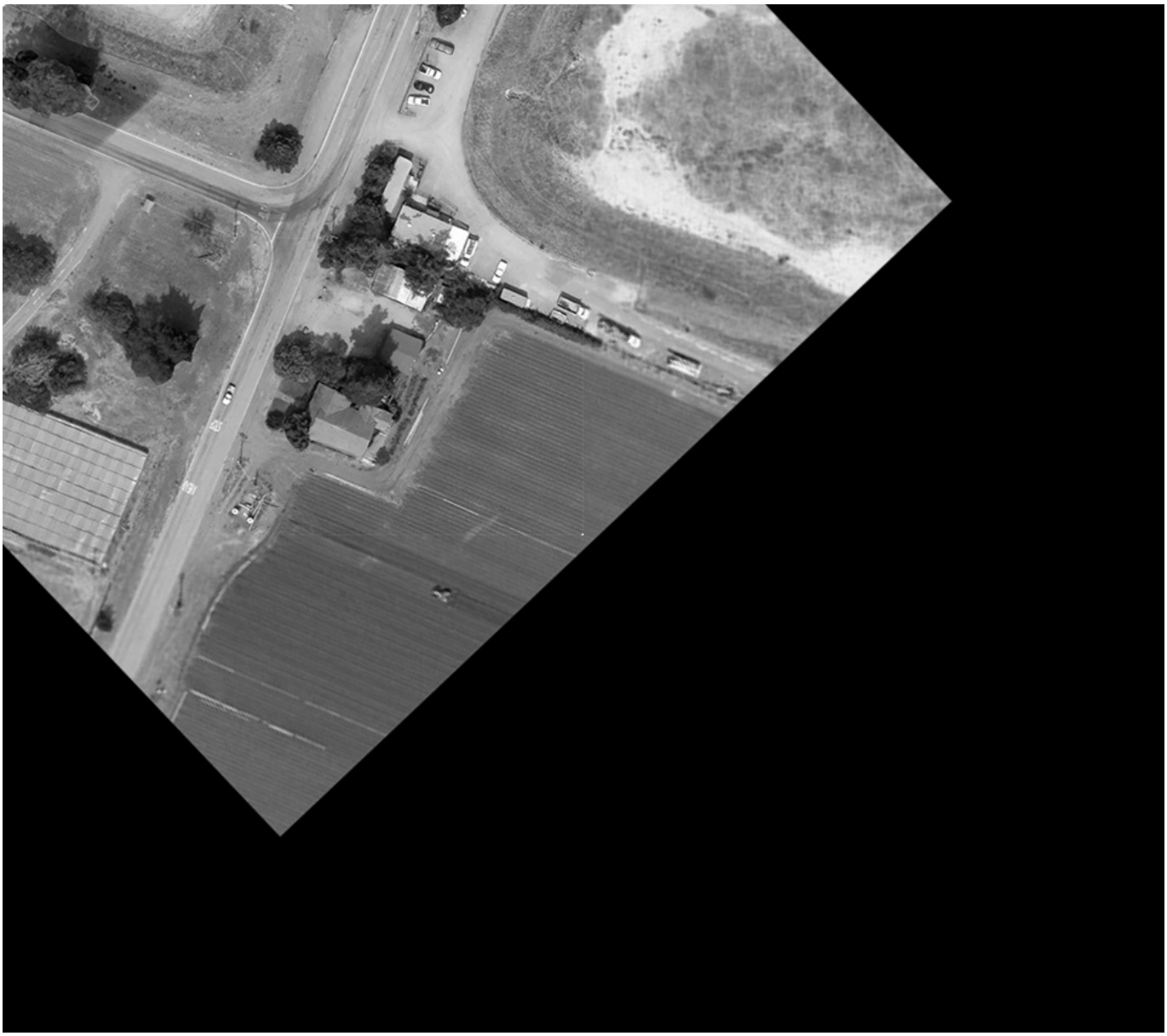

Imagem Panorâmica - SIFT com matches com distância de no máximo 200

Porcentagem

```
Ratio: 0.7
Matches: 344
```

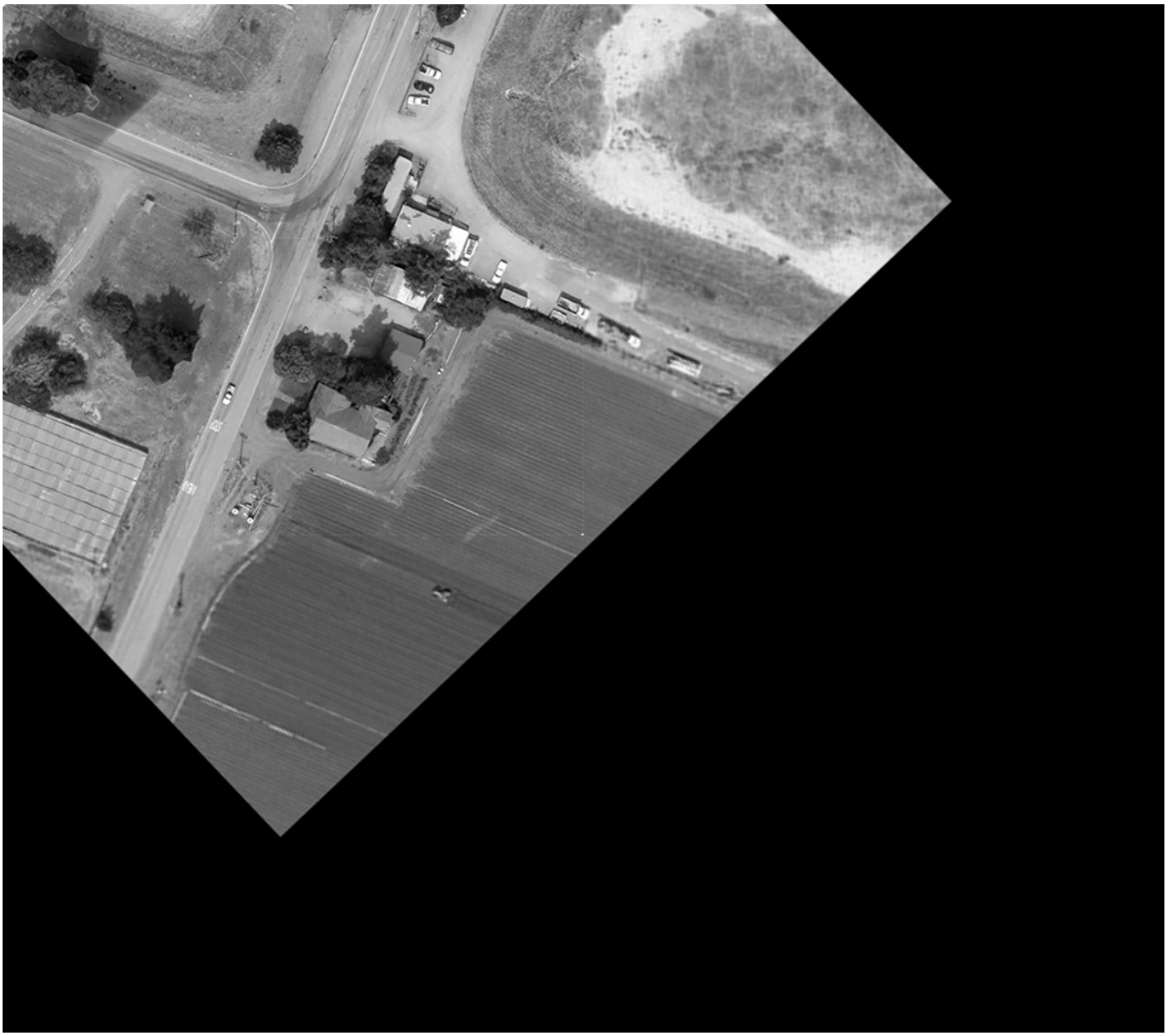

Imagem Panorâmica - SIFT com matches com ratio de distância de 0.7

```
Ratio: 0.4
Matches: 278
```

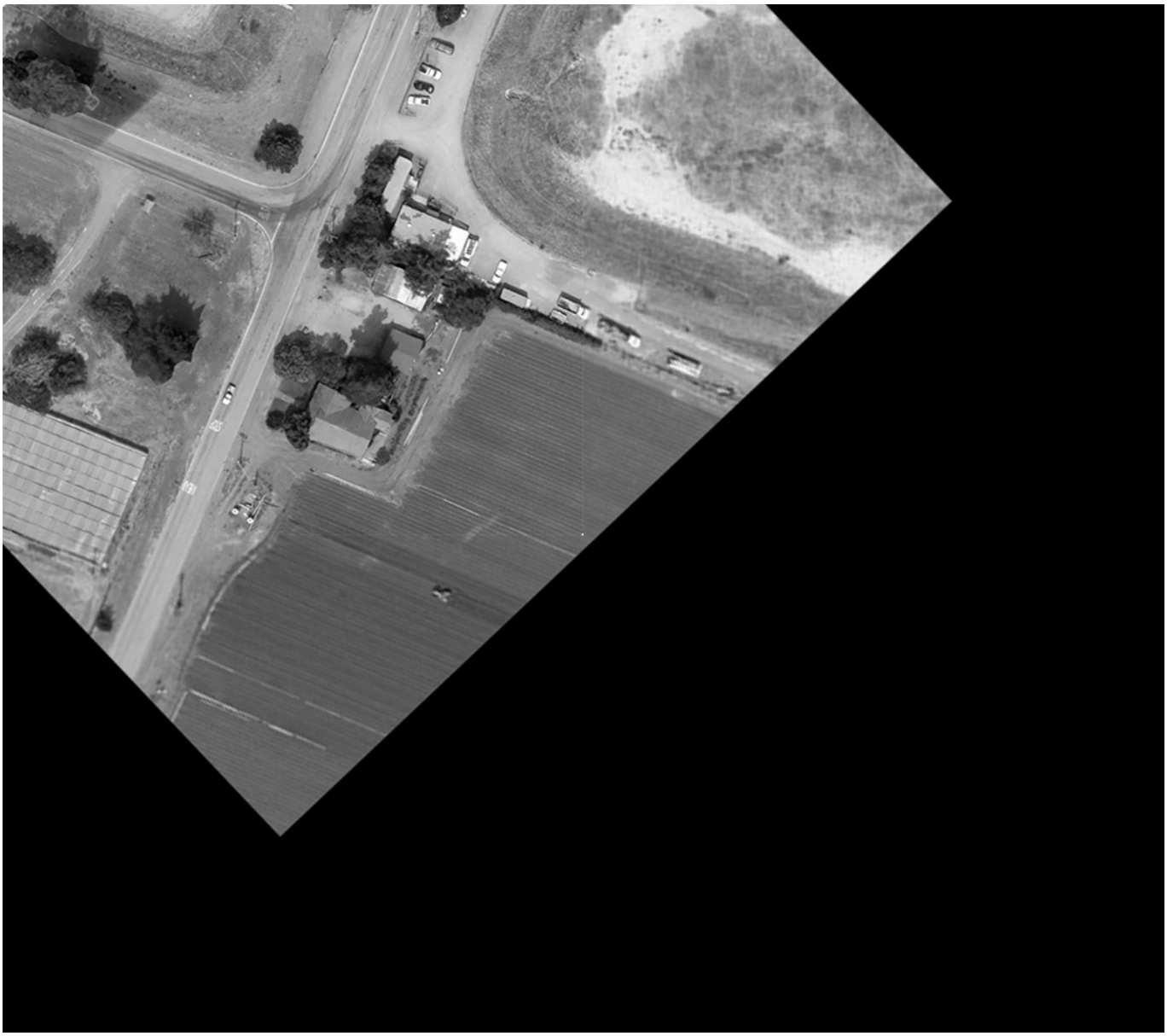

Imagem Panorâmica - SIFT com matches com ratio de distância de 0.4

```
Ratio: 0.9
Matches: 532
```

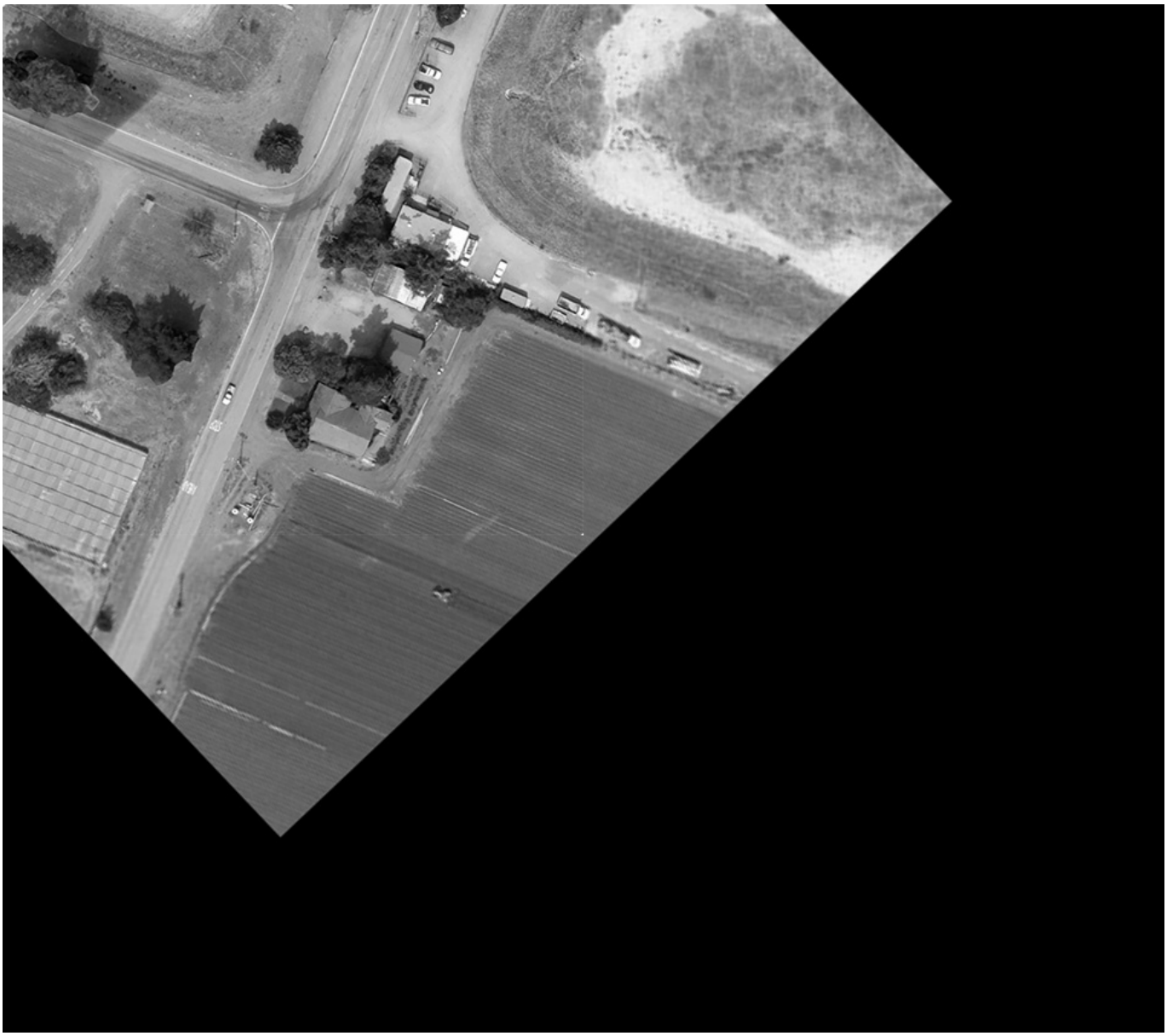

Imagem Panorâmica - SIFT com matches com ratio de distância de 0.9

Como era de se esperar, quanto menor o *threshold*, menos matches serão obtidos. Estes matches, porém, representam com fidelidade a correspondência entre as imagens e a sua escolha deve ser feita pensando-se nesse equilíbrio entre resultado final e performance.

Referências

1. Documentação das bibliotecas usadas
    - OpenCV
    - NumPy
    - Jupyter Notebook
2. *R.C. Gonzalez, R.E. Woods. Digital Image Processing. Prentice Hall, 2007*.
3. Material de aula fornecido pelo Professor