Universidade Federal do Paraná
Setor de Ciências Exatas
Departamento de Informática

Vitória Stavis de Araujo

Análise comparativa de técnicas de remoção de ruído com OpenCV

Trabalho apresentado para obtenção de nota parcial na disciplina Processamento de Imagens, no quinto período do curso de Informática Biomédica, ministrada pelo professor Luiz Oliveira.

Curitiba
2022

1 INTRODUÇÃO

Ruídos são variações aleatórias presentes em qualquer aparelho eletrônico que transmite e recebe sinais. No caso das imagens, aparecem como pontos aleatórios e podem diminuir a qualidade da imagem. Os ruídos também podem ser aplicados por meio de algoritmos e programas de edição de imagem e, dependendo do uso, podem melhorar a nitidez da imagem. Computacionalmente, os ruídos são representados por transformações nos valores dos pixels.

Para remover ruídos computacionalmente, existem várias técnicas; alguns métodos de filtragem foram avaliados neste trabalho.

O tipo de ruído gerado nas imagens para o trabalho foi o salt and pepper, em que pixels são aleatoriamente modificados para o valor mínimo ou máximo, gerando pontos brancos e pretos.

2 METODOLOGIA

Foi usada a linguagem de programação Python e as bibliotecas opencv para processar as imagens, matplotlib para visualizar dados e numpy para operações.

O script filtro.py gera uma imagem com ruído salt and pepper, a partir de uma imagem original, e usa um dos seguintes métodos para removê-lo:

- Filtro da média por filter2D, que usa uma matriz kernel determinada para filtrar a imagem, tira a média e substitui o pixel central pelo novo valor médio.
- Filtro da média com blur, que borra a imagem a partir de um box filter normalizado com tamanho determinado.
- Filtro da média com blur gaussiano, que borra a imagem da mesma forma do blur, mas seguindo distribuição gaussiana, determinando tamanho e desvio padrão.
- Filtro da mediana, que substitui o pixel pelo valor da mediana dos seus vizinhos.
- Empilhamento de imagens, em que são sobrepostas várias imagens ruidosas e depois cada pixel é dividido pelo número de imagens.

O script tem como parâmetros a imagem original, a intensidade do ruído salt and pepper, o filtro utilizado para remover o ruído e o nome do arquivo de saída.

Além disso, o script tests.py possui funções que foram utilizadas para testar o melhor método de remoção do filtro. Para isso, foram geradas 5 imagens com ruído salt and pepper de intensidades diferentes e foram testados todos os métodos. Para comparação do resultado, foi utilizada a função cv.PSNR, que compara a imagem original (sem ruído) com a imagem com ruído removido. Um valor é gerado e, quanto maior o valor, mais próximas as imagens são.

Argumentos testados para cada método:

- Average_2D:

```
kernel = np.ones((15, 15), np.float32)/255

kernel = np.ones((17, 17), np.float32)/255

kernel = np.ones((19, 19), np.float32)/255
```

- Average_blur:

```
res = cv.blur(img, ksize=(2, 2))

res = cv.blur(img, ksize=(4, 4))

res = cv.blur(img, ksize=(6, 6))
```

- Average_gaussian:

```
res = cv.GaussianBlur(src = img, ksize =(3, 3), sigmaX = 0)

res = cv.GaussianBlur(src = img, ksize = (5, 5), sigmaX = 0)

res = cv.GaussianBlur(src = img, ksize = (7, 7), sigmaX = 0)
```

- Median:

```
res = cv.medianBlur(img, 3)

res = cv.medianBlur(img, 5)

res = cv.medianBlur(img, 7)
```

- Stacking

```
   15, 25, 50, 100, 150, 500 imagens empilhadas
```

3 RESULTADOS

3.1 Testes

----- Average Filter 2D tests ------
Noise: 0.01 Average 2D 15, PSNR: 19.162
Noise: 0.01 Average 2D 17, PSNR: 18.994
Noise: 0.01 Average 2D 19, PSNR: 13.922

Noise: 0.02 Average 2D 15, PSNR: 18.993
Noise: 0.02 Average 2D 17, PSNR: 19.136
Noise: 0.02 Average 2D 19, PSNR: 13.898

Noise: 0.05 Average 2D 15, PSNR: 18.422
Noise: 0.05 Average 2D 17, PSNR: 19.487
Noise: 0.05 Average 2D 19, PSNR: 13.813

Noise: 0.07 Average 2D 15, PSNR: 18.045
Noise: 0.07 Average 2D 17, PSNR: 19.568
Noise: 0.07 Average 2D 19, PSNR: 13.725

Noise: 0.1 Average 2D 15, PSNR: 17.44
Noise: 0.1 Average 2D 17, PSNR: 19.586
Noise: 0.1 Average 2D 19, PSNR: 13.625

Para o filtro de média com filter2D, o tamanho de kernel 15 foi melhor para o filtro de nível 0.01. Para níveis maiores (>= 0.05), o tamanho 17 foi melhor.

----- Average Blur tests ------
Noise: 0.01 Average blur 2, PSNR: 24.527
Noise: 0.01 Average blur 4, PSNR: 24.422
Noise: 0.01 Average blur 6, PSNR: 23.583

Noise: 0.02 Average blur 2, PSNR: 22.845
Noise: 0.02 Average blur 4, PSNR: 23.857
Noise: 0.02 Average blur 6, PSNR: 23.314

Noise: 0.05 Average blur 2, PSNR: 19.807
Noise: 0.05 Average blur 4, PSNR: 22.294
Noise: 0.05 Average blur 6, PSNR: 22.375

Noise: 0.07 Average blur 2, PSNR: 18.545
Noise: 0.07 Average blur 4, PSNR: 21.396
Noise: 0.07 Average blur 6, PSNR: 21.715

Noise: 0.1 Average blur 2, PSNR: 17.025
Noise: 0.1 Average blur 4, PSNR: 20.098
Noise: 0.1 Average blur 6, PSNR: 20.643

Para o filtro de média com a função blur, o parâmetro 2 foi melhor para o filtro de nível 0.01. Para níveis maiores (>= 0.05), o tamanho 4 foi melhor.

----- Average Gaussian Blur tests ------
Noise: 0.01 Average gaussian 3, PSNR: 26.517
Noise: 0.01 Average gaussian 5, PSNR: 25.775
Noise: 0.01 Average gaussian 7, PSNR: 24.86

Noise: 0.02 Average gaussian 3, PSNR: 24.922
Noise: 0.02 Average gaussian 5, PSNR: 24.918
Noise: 0.02 Average gaussian 7, PSNR: 24.404

Noise: 0.05 Average gaussian 3, PSNR: 21.893
Noise: 0.05 Average gaussian 5, PSNR: 22.799
Noise: 0.05 Average gaussian 7, PSNR: 23.005

Noise: 0.07 Average gaussian 3, PSNR: 20.531
Noise: 0.07 Average gaussian 5, PSNR: 21.652
Noise: 0.07 Average gaussian 7, PSNR: 22.101

Noise: 0.1 Average gaussian 3, PSNR: 18.891
Noise: 0.1 Average gaussian 5, PSNR: 20.143
Noise: 0.1 Average gaussian 7, PSNR: 20.78

Para o filtro de média com gaussian blur, o parâmetro 3 foi o melhor nos níveis mais baixos de ruído (0.01 e 0.02) e, a partir do nível 0.05, o tamanho 7 foi melhor. Dos métodos de média, o gaussian blur foi o que teve melhores resultados, então ficará como padrão no script filtro.py, caso seja passado o parâmetro 0 (média) para remover o ruído.

----- Median filter tests ------
Noise: 0.01 Median 3, PSNR: 27.242
Noise: 0.01 Median 5, PSNR: 24.588
Noise: 0.01 Median 7, PSNR: 23.701

Noise: 0.02 Median 3, PSNR: 27.129
Noise: 0.02 Median 5, PSNR: 24.559
Noise: 0.02 Median 7, PSNR: 23.689

Noise: 0.05 Median 3, PSNR: 26.742
Noise: 0.05 Median 5, PSNR: 24.475
Noise: 0.05 Median 7, PSNR: 23.647

Noise: 0.07 Median 3, PSNR: 26.282
Noise: 0.07 Median 5, PSNR: 24.413
Noise: 0.07 Median 7, PSNR: 23.617

Noise: 0.1 Median 3, PSNR: 25.199
Noise: 0.1 Median 5, PSNR: 24.296
Noise: 0.1 Median 7, PSNR: 23.564

Para o filtro da mediana, o tamanho 3 foi melhor para todos os níveis de ruído.

----- Image stacking tests -----
Noise: 0.01 Stacking 15, PSNR: 33.383
Noise: 0.01 Stacking 25, PSNR: 35.397
Noise: 0.01 Stacking 50, PSNR: 37.793
Noise: 0.01 Stacking 100, PSNR: 39.904
Noise: 0.01 Stacking 150, PSNR: 40.98

Noise: 0.02 Stacking 15, PSNR: 30.178
Noise: 0.02 Stacking 25, PSNR: 32.012
Noise: 0.02 Stacking 50, PSNR: 34.2
Noise: 0.02 Stacking 100, PSNR: 35.999
Noise: 0.02 Stacking 150, PSNR: 36.796

Noise: 0.05 Stacking 15, PSNR: 25.548
Noise: 0.05 Stacking 25, PSNR: 27.081
Noise: 0.05 Stacking 50, PSNR: 28.73
Noise: 0.05 Stacking 100, PSNR: 29.883
Noise: 0.05 Stacking 150, PSNR: 30.347

Noise: 0.07 Stacking 15, PSNR: 23.704
Noise: 0.07 Stacking 25, PSNR: 25.079
Noise: 0.07 Stacking 50, PSNR: 26.485
Noise: 0.07 Stacking 100, PSNR: 27.406
Noise: 0.07 Stacking 150, PSNR: 27.765

Noise: 0.1 Stacking 15, PSNR: 21.635
Noise: 0.1 Stacking 25, PSNR: 22.82
Noise: 0.1 Stacking 50, PSNR: 23.963
Noise: 0.1 Stacking 100, PSNR: 24.675
Noise: 0.1 Stacking 150, PSNR: 24.952

Resultados com 500 imagens

Noise: 0.01 Stacking 500, PSNR: 42.948
Noise: 0.02 Stacking 500, PSNR: 38.219
Noise: 0.05 Stacking 500, PSNR: 31.078
Noise: 0.07 Stacking 500, PSNR: 28.323
Noise: 0.1 Stacking 500, PSNR: 25.566

O método do empilhamento foi o melhor método, sendo 150 imagens o parâmetro com os melhores resultados. Aumentar muito o número de imagens não faz diferença relevante e aumenta significativamente o tempo de execução.

4 CONCLUSÕES

Conclui-se que o melhor método, de acordo com o PSNR, é o de empilhamento de imagens, para todos os níveis de ruído. Apesar disso, ele demora muito para executar, com o tempo aumentando muito de acordo com o número de imagens empilhadas. O método da mediana alcançou PSNR acima de 25 para todos os níveis de ruído e executa muito mais rápido.

5 REFERÊNCIAS

https://www.cambridgeincolour.com/pt-br/tutoriais/image-noise.htm
https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html