

Universidade Federal de Alagoas
Instituto de Computação

Processamento Digital de Imagens
Professor Tiago Vieira

Hiago Cavalcante e Luana Ferreira
Reprodução dos Resultados Obtidos

Sumário

1	Introdução	3
2	Resumo do artigo	3
3	Outline do código-fonte	5
3.1	Scripts	5
3.2	Diagrama de Classes	6
4	Reprodução dos resultados	6
4.1	Ruído Gaussiano	7
4.2	Ruído Sal e Pimenta	7
4.3	Ruído de Poisson	8
4.4	Ruído Speckle	8
4.5	Adição de múltiplos filtros	8
4.6	Resultados obtidos com a remoção de ruídos	9
4.7	Resultados obtidos com a remoção de ruídos utilizando COBRA	9
5	Discussão acerca dos resultados obtidos	9
6	Conclusões e trabalhos futuros	10

1 Introdução

Esse relatório contém a reprodução dos resultados obtidos pelos autores do artigo *Non-linear aggregation of filters to improve image denoising* como parte do trabalho que constitui a disciplina de Processamento Digital de Imagens, ministrada pelo professor Tiago Vieira. Esse projeto está em desenvolvimento e encontra-se em

<https://github.com/ferreiraluana/dip-project>. O artigo escolhido está disponível na web em: <https://paperswithcode.com/paper/non-linear-aggregation-of-filters-to-improve>.

2 Resumo do artigo

A remoção de ruídos é um processo fundamental no tratamento de imagens. Diversos fatores podem influenciar no tipo e na qualidade dos ruídos presentes nas imagens, tais quais a iluminação excessiva ou a falta de iluminação, falta de ajuste na profundidade do campo visual, o desfoque, entre outros. Além disso, fatores como o aumento da temperatura do sensor, o erro de transmissão de dados e as aproximações feitas durante a digitalização também contribuem para a adição de ruído. Nesse processo, o principal desafio é remover o ruído sem comprometer a estrutura base componente da imagem.

Os autores [1] deste trabalho apresentam uma abordagem inusitada para a remoção de ruído em imagens. A ideia consiste em combinar diferentes métodos clássicos de remoção de ruídos com o objetivo de obter várias previsões para um mesmo pixel, obtendo então, a eliminação do ruído. Cada método clássico de remoção de ruído possui pontos fortes e fracos. Sendo assim, os pesquisadores decidiram implementar nesse artigo uma forma de aproveitar o melhor de cada método clássico. Para isso, foi utilizado o algoritmo COBRA - Combined Regression Alternative, disponível na biblioteca PYCOBRA/Python.

Essa estratégia de agregação de vários filtros de remoção de ruído é conhecida na academia como Filtragem Colaborativa. Um algoritmo de Filtragem Colaborativa muito conhecido e utilizado como base para os autores é o BM3D (*block-matching and 3D collaborative filtering*), o qual une pedaços das imagens 2D, as quais são fragmentos das imagens 3D e produzem uma estimativa ao unirmos os blocos filtrados.

O método utilizado pelos autores é explicado na figura 1:

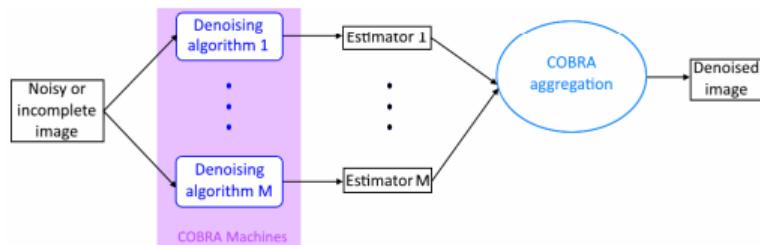


Figura 1: Modelo geral para a versão do algoritmo COBRA utilizado na remoção de ruído. Fonte: Guedj and Rengot, 2020

O algoritmo utilizado pelos autores é descrito a seguir em 2:

Algorithm 1 Image denoising with COBRA aggregation

INPUT:

im_noise = the noisy image to denoise
p_size = the pixel patch size to consider

M = the number of COBRA machines to use

OUTPUT:

Y = the denoised image

```

Xtrain  $\leftarrow$  training images with artificial noise
Ytrain  $\leftarrow$  original training images (ground truth)
cobra  $\leftarrow$  initial COBRA model
cobra  $\leftarrow$  to adjust COBRA model parameters with respect to the data (Xtrain,
Ytrain)
cobra  $\leftarrow$  to load M COBRA machines
cobra  $\leftarrow$  to aggregate the predictions
Xtest  $\leftarrow$  feature extraction from im_noise in a vector of size (nb_pixels,  $(2 \cdot p_{size} + 1)^2$ )
Y  $\leftarrow$  prediction of Xtest by cobra
Y  $\leftarrow$  to add im_noise values lost at the borders of the image, because of the patch
processing, to Y

```

Figura 2: Algoritmo para remoção de ruídos em imagens com agregação COBRA. Fonte: Guedj and Rengot, 2020

Os ruídos clássicos que os autores decidiram agregar e adicionar às imagens em conformidade com o algoritmo COBRA são descritos em 3:



Figura 3: Diferentes tipos de ruídos utilizados nos experimentos. Fonte: Guedj and Rengot, 2020

Para o treinamento, foram utilizadas 25 imagens definidas como base, ou seja, sem a presença de ruídos. Então, foram adicionados os diferentes ruídos apresentados em 3, produzindo 125 imagens ruidosas. A partir daí, duas cópias independentes de cada uma dessas imagens ruidosas foram criadas adicionando o filtro normal. Uma das imagens foi direcionada ao *data pool* para o treinamento preliminar dos filtros, enquanto que a outra imagens ruidosa vai para o *data pool* com o fito de computar os pesos definidos pelas equações do modelo. A construção do *dataset* é ilustrada em 4:

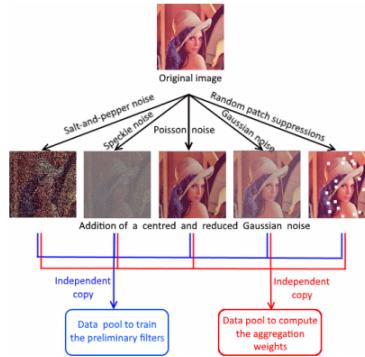


Figura 4: Construção do Dataset. Fonte: Guedj and Rengot, 2020

Há dois meta-parâmetros no COBRA que são otimizados utilizando o *cross-validation*: $\alpha = 4/7$ e $\epsilon = 0, 2$. Já para a avaliação da performance do algoritmo, foram utilizadas quatro métricas: *Mean Absolute Error* (MAE), *Root Mean Square Error* (RMSE), *Peak Signal to Noise Ratio* (PSNR), *Universal image Quality Index* (UQI). Para computar a estatística descritiva, os experimentos foram repetidos 100 vezes e o algoritmo COBRA atingiu os melhores resultados na remoção de ruídos em todos os filtros testados de acordo com as métricas calculadas em 5:

Denoising method	MAE		RMSE		PSNR		UQI	
	Average	std	Average	std	Average	std	Average	std
COBRA	unknown noise	0.0802	0.0041	0.1155	0.0050	66.8771	0.2741	0.9557
	known noise	0.0801	0.0039	0.1083	0.0048	67.7980	0.2699	0.9601
	<i>Bilateral filter</i>	0.1074	0.0058	0.1786	0.0082	63.0962	0.4001	0.9139
	<i>Non-local means</i>	0.1154	0.0062	0.1844	0.0091	62.8274	0.4302	0.9038
	<i>Gaussian filter</i>	0.1051	0.0048	0.1873	0.0076	62.6849	0.3536	0.9109
	<i>Median filter</i>	0.1334	0.0049	0.1769	0.0058	63.1068	0.2333	0.8991
	<i>TV-Chambolle</i>	0.1124	0.0056	0.1616	0.0070	63.8476	0.3609	0.9190
	<i>Richardson-Lucy</i>	0.1741	0.0033	0.2294	0.0036	60.9190	0.1415	0.8940
	<i>Inpainting</i>	0.0898	0.0045	0.1576	0.0069	64.2497	0.3682	0.9398
	<i>K-SVD</i>	0.1096	0.0055	0.1623	0.0070	63.9382	0.3702	0.9208
	BM3D	0.1024	0.0029	0.1392	0.0105	65.2288	0.0399	0.9378
	<i>Lee</i>	0.1369	0.0036	0.1778	0.0048	62.9373	0.2711	0.9001
								0.0046

Figura 5: Métricas obtidas na remoção de ruído de uma imagem adicionada de vários tipos de ruídos. Fonte: Guedj and Rengot, 2020

3 Outline do código-fonte

3.1 Scripts

O código-fonte está organizado da seguinte forma:

- **noise.py**: contém um módulo para adicionar ruídos artificiais a imagens (Gaussian, Poisson, sal e pimenta, speckle, supressão aleatória, multi);
- **denoise.py**: contém um módulo para eliminar o ruído de imagens com métodos clássicos (filtro Gaussiano, filtro Mediano, filtro bilateral, meios não locais, TV-Chambolle, deconvolução Richardson-Lucy, inpainting);
- **denoising_cobra.py**: contém um módulo para avaliar a qualidade de denoising (RMSE, PSNR);
- **evaluation.py**: contém o arquivo principal para criar um modelo cobra e usá-lo para a tarefa de remoção de ruído.

3.2 Diagrama de Classes

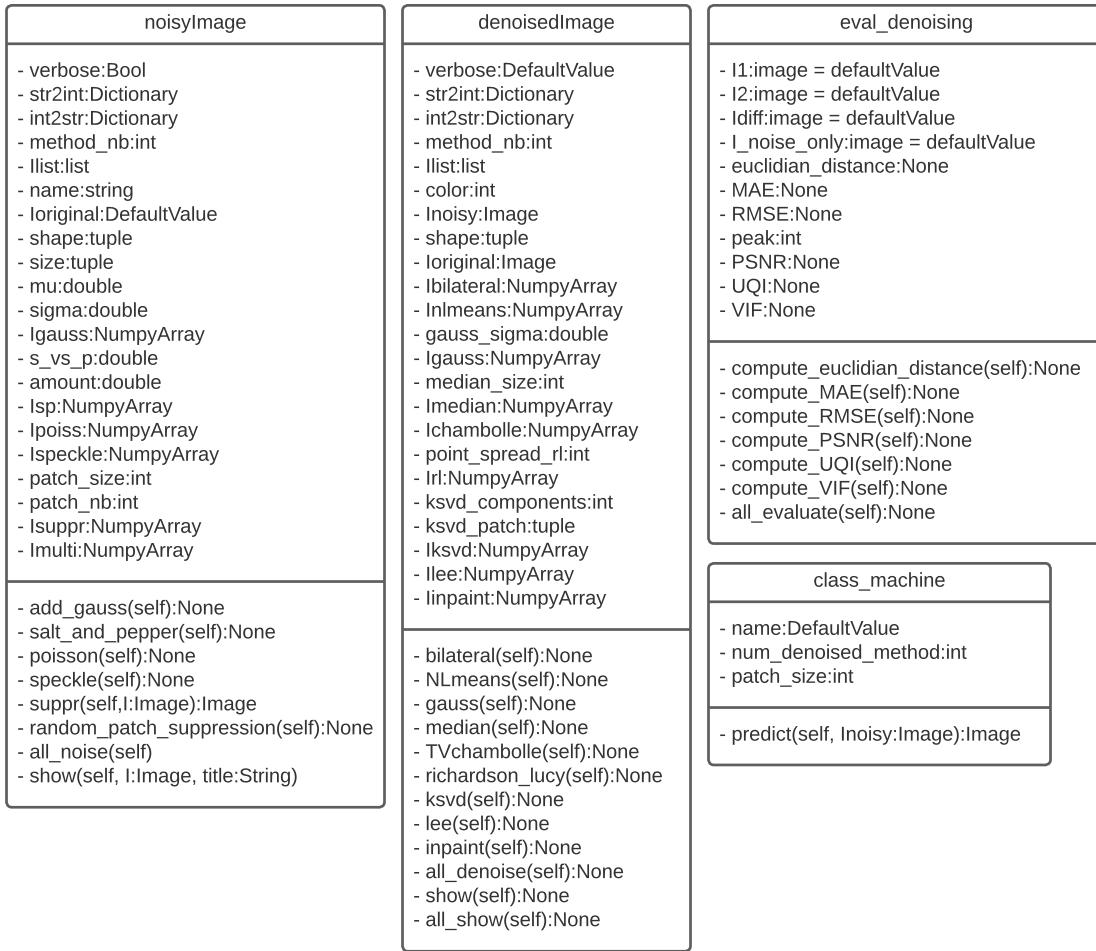


Figura 6: Diagrama de Classes UML. Fonte: autores, 2022.

4 Reprodução dos resultados

Para a reprodução dos resultados, foi utilizada a IDE Visual Studio Code e o Google Colaboratory. O código em Python facilita a leitura e a compreensão do algoritmo, além do fato de a documentação do projeto no GitHub explicar em detalhes como utilizar os scripts.

A principal dificuldade encontrada durante a execução dos scripts foi a insuficiência de memória RAM, pois o algoritmo requer uma memória RAM maior que 12,6GB, o que excede até o Google Colaboratory.

Foram utilizadas 25 imagens do tipo PNG para o treinamento do algoritmo e as bibliotecas necessárias para o funcionamento do projeto são:

- OpenCV;
- Numpy;

- Scikit-image;
- Pycobra;
- KSVD;
- Scikit-learn
- Scipy
- Matplotlib

A primeira etapa do projeto é a adição de ruídos por meio de ruídos classicamente conhecidos na literatura, os quais serão reproduzidos a seguir.

4.1 Ruído Gaussiano



Figura 7: Adição de ruído gaussiano. Fonte: autores, 2022.

4.2 Ruído Sal e Pimenta



Figura 8: Adição de ruído Sal e Pimenta. Fonte: autores, 2022.

4.3 Ruído de Poisson



Figura 9: Adição de ruído de Poisson. Fonte: autores, 2022.

4.4 Ruído Speckle



Figura 10: Adição de ruído Speckle. Fonte: autores, 2022.

4.5 Adição de múltiplos filtros

Por fim, os autores aplicam diversos filtros simultaneamente à mesma imagem:

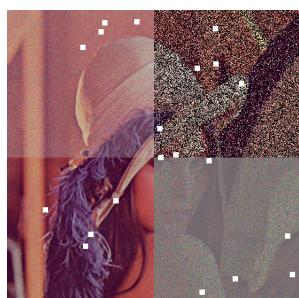


Figura 11: Adição de múltiplos ruídos. Fonte: autores, 2022.

E ainda aplicam um ruído que retira partes aleatórias da mesma imagem:

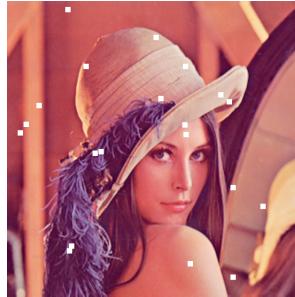


Figura 12: Adição de ruído com perda de informação. Fonte: autores, 2022.

4.6 Resultados obtidos com a remoção de ruídos

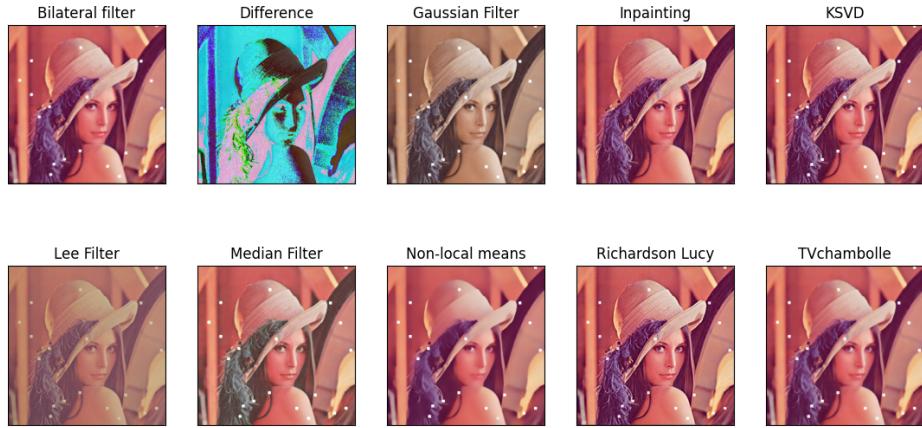


Figura 13: Resultados reproduzidos da remoção de ruídos clássica. Fonte: autores, 2022.

4.7 Resultados obtidos com a remoção de ruídos utilizando COBRA

Não foi possível obter os resultados do último script por motivos de insuficiência de memória RAM e tempo excessivo de treinamento.

5 Discussão acerca dos resultados obtidos

Observamos que os resultados obtidos são compatíveis com os resultados dos autores Guedj e Rengot (2020). Esses autores comprovaram que o filtro mediano é mais adequado para o ruído sal e pimenta, por exemplo, mas o algoritmo COBRA supera a performance de remoção de ruídos quando há vários tipos de ruído, o que foi demonstrado com as métricas explicitadas em (5).

6 Conclusões e trabalhos futuros

Sendo assim, os autores concluem que em uma abordagem mais genérica no sentido de que quaisquer filtros preliminares podem ser agregados, independentemente de sua natureza e habilidades específicas, experimentos numéricos sugerem que o algoritmo COBRA atinge o melhor desempenho ao lidar com diferentes tipos de ruídos, pois possui a capacidade de se adaptar aos diferentes tipos de ruídos.

Como trabalhos futuros, pretende-se otimizar o código de modo a conseguir resultados do script referente ao algoritmo COBRA e comparar os resultados com o trabalho original. Além disso, propõe-se identificar quais as possíveis razões para os autores decidirem não utilizar o filtro BM3D com um estudo de caso sobre este.

Referências

- [1] Benjamin Guedj and Juliette Rengot. Non-linear aggregation of filters to improve image denoising. In *Science and Information Conference*, pages 314–327. Springer, 2020.