scc0251/5830— Prof. Moacir Ponti

PAE/monitor: Fernando Pereira dos Santos, Eduardo Santos Carlos Souza

Trabalho 6: restauração de imagens

Desenvolva seu código sem olhar o de colegas. Plágio não será tolerado.

Nesse trabalho você deverá implementar três filtros de restauração de imagens.

Você deverá implementar usando `python 3` com as bibliotecas `numpy` e `imageio`.

Leia com calma as instruções de cada etapa.

Seu programa deverá permitir receber o nome de uma imagem ruidosa, a partir da qual você irá aplicar um dos três métodos de filtragem adaptativo escolhidos e comparar com a imagem original sem ruídos.

Entrada

1. **Receber via teclado**:
   a) nome da imagem original para comparação (*Icomp*);
   b) nome da imagem ruidosa para filtragem (*Inoisy*);
   c) opção de escolha do método (1 – filtro adaptativo de redução de ruído local, 2 – filtro adaptativo da mediana, 3 – filtro da média contra-harmônica);
   d) tamanho do filtro (N): $n \times n$
   e) parâmetros específicos para os filtros:
      – para o método 1: o valor da distribuição de ruído (σ_{noisy})
      – para o método 2: o tamanho máximo do filtro (M): $m \times m$
      – para o método 3: a ordem do filtro (Q)

Observação 1: Os filtros serão sempre quadráticos. Por exemplo, se N é 2, o filtro será 2×2. O mesmo vale para M, sendo que $M >= N$.

Métodos de restauração de imagens

Opção 1: Filtro adaptativo de redução de ruído local

Neste método de restauração, utilizado com ruídos gaussianos, o filtro (N) é definido de acordo com a vizinhança do ponto central considerado. Portanto, o valor resultante para cada ponto é calculado baseado na variância do ruído da imagem (${\sigma^2}_{noisy}$), na variância local dos pixels contidos na delimitação da vizinhança do filtro (${\sigma^2}_N$) e na média dos pixels contidos no filtro (m_N):

$$Iout = Inoisy - \frac{{\sigma^2}_{noisy}}{{\sigma^2}_N} \cdot [Inoisy - m_N]$$

Observação 2: Para aplicar a vizinhança nos pixels da margem, você deverá considerar que a imagem é uma matriz circular (image wrap). Como apresentado a seguir:

00	01	02	03	04
10	11	12	13	14
20	21	22	23	24
30	31	32	33	34
40	41	42	43	44

(a) Imagem ruidosa de entrada

44	40	41	42	43	44	40
04	00	01	02	03	04	00
14	10	11	12	13	14	10
24	20	21	22	23	24	20
34	30	31	32	33	34	30
44	40	41	42	43	44	40
04	00	01	02	03	04	00

(b) Imagem ruidosa para filtragem

Observação 3: Lembre-se que a quantidade de linhas e colunas utilizadas no image wrap depende diretamente do tamanho do filtro (N). Isso vale para todos os três filtros aqui descritos.

Opção 2: Filtro adaptativo de mediana

Sendo utilizado com ruídos impulsivos (sal & pimenta), o filtro adaptativo da mediana é constituído por duas etapas distintas (A e B). O filtro utilizado considera uma vizinhança de tamanho N ($n \times n$), considerando que: z_{min} é o valor mínimo existente na vizinhança; z_{max} é o valor máximo existente na vizinhança; z_{med} é o valor da mediana existente na vizinhança; e M ($m \times m$) é o tamanho máximo do filtro. Portanto, os passos do método para cada pixel (x, y) são:

1. ETAPA A:

   A1 = z_{med} - z_{min}

   A2 = z_{med} - z_{max}

   SE (A1 $>$ 0 .E. A2 $<$ 0) ETAPA B

   SENÃO

   Aumente o tamanho do filtro (vizinhança) em 1: $N_{atual} = (n + 1\ X\ n + 1)$

   SE ($N_{atual} <= M$) ETAPA A

   SENÃO Iout(x,y) = z_{med}

2. ETAPA B:

   B1 = $Inoisy(x, y)$ - z_{min}

   B2 = z_{med} - z_{max}

SE (B1 $>$ 0 .E. B2 $<$ 0) $Iout(x, y) = Iout(x, y)$

SENÃO $Iout(x, y) = z_{med}$

Observação 4: O aumento no tamanho do filtro (N) é linear até o seu tamanho máximo (M). Por exemplo, considerando que o tamanho inicial seja $n = 3$ temos uma vizinhança 3×3. Com o aumento será 4×4, depois 5×5 e, assim, sucessivamente até $(n = m)$.

Observação 5: Para aplicar a vizinhança nos pixels da margem, você deverá replicar os próprios valores. Como apresentado a seguir:

00	01	02	03	04
10	11	12	13	14
20	21	22	23	24
30	31	32	33	34
40	41	42	43	44

(a) Imagem ruidosa de entrada

00	00	01	02	03	04	04
00	00	01	02	03	04	04
10	10	11	12	13	14	14
20	20	21	22	23	24	24
30	30	31	32	33	34	34
40	40	41	42	43	44	44
40	40	41	42	43	44	44

(b) Imagem ruidosa para filtragem

Opção 3: Filtro da média contra-harmônica

O filtro da média contra-harmônica também é utilizado com ruídos impulsivos, porém funciona somente com remoção de pixels brancos ou pretos, nunca simultaneamente. Para este método, a ordem do filtro (Q) é um parâmetro necessário para eliminar ruídos do tipo sal ($Q < 0$) ou do tipo pimenta ($Q > 0$), sendo que (g) é a região de ($Inoisy$) delimitada pelo filtro com centro em *(x,y)*.

$$Iout = \frac{\sum g^{Q+1}}{\sum g^{Q}}$$

Observação 6: Para fazer aplicar esse método considere o preenchimento pelo valor zero em todas as posições fora da margem.

Comparação com imagem de referência

Após gerar a imagem de saída $Iout$, comparar com a imagem original de entrada $Icomp$ utilizando o RMSE:

$$RMSE = \sqrt{\frac{1}{RS} \sum_i \sum_j (Icomp(i, j) - Iout(i, j))^2},$$

onde $R \times S$ são as resoluções das imagens sendo comparadas.

Saída

A saída será o valor do RMSE.

Exemplos de entrada e saída

Exemplo de entrada 1: imagem de comparação, imagem ruidosa, método de restauração, tamanho do filtro e valor da distribuição de ruído:

```
dog_original.png
dog_noisy.png
1
5
0.025
```

Exemplo de entrada 2: imagem de comparação, imagem ruidosa, método de restauração, tamanho do filtro e tamanho máximo do filtro:

```
dog_original.png
dog_noisy.png
2
3
7
```

Exemplo de entrada 3: imagem de comparação, imagem ruidosa, método de restauração, tamanho do filtro e ordem do filtro:

```
dog_original.png
dog_noisy.png
3
3
-1.3
```

Exemplos de saída: RMSE

```
1.5692
```

Submissão e instruções

No sistema Run.Codes deve incluir apenas o arquivo `.py`

1. **É obrigatório comentar seu código**. Como cabeçalho insira seu nome, número USP, código da disciplina, ano/semestre e o título do trabalho. Haverá desconto na nota caso esse cabeçalho esteja faltando, além de descontos para falta de comentários.

2. **É obrigatório organizar seu código em funções**.