†

UFABC	**Universidade Federal do ABC** **Bacharelado em Ciência da Computação** **Disciplina:** Processamento Digital de Imagens **Prof.:** Francisco Zampirolli **Turma:** Imagem 2020 **Sala:** 123 **Exame:** Atividade 6 **Data:** 31-05-2020
Ass.: _____ **Estudante:** Marcelo Pena **ID/RA:** 11039314	

#138 - 2020-05-25 - 12:06:42

Instruções:

(a) Esta é a Atividade 6 do ECE, para ser enviada pelo Moodle.
(b) Esta é uma atividade individual, foram geradas mais de 50 variações desta questão e cada aluno vai receber uma questão distinta.
(c) Sugestão: resolver o problema primeiro para dimensões pequenas, para facilitar a validação do seu código.
(d) Antes de submeter, valide o seu código em IDE's com Jupyter Notebook, ou `https://repl.it/languages/python3`

Questões Dissertativas:

1. Considere a dilatação definida por:

$$dil(f,b) = f \oplus b = \delta_b(f) = \max\{f(y) + b(y-x) : y \in \mathbb{B}_x \cap \mathbb{E}, \forall x \in \mathbb{E}\}$$

Considere a erosão definida por:

$$ero(f,b) = f \ominus b = \varepsilon_b(f) = \min\{f(y) - b(x-y) : y \in \mathbb{B}_x \cap \mathbb{E}, \forall x \in \mathbb{E}\}$$

Onde $f \in K^{\mathbb{E}}$ ou $f \in [0,k]^{\mathbb{E}}$, k é um inteiro positivo representando os níveis de cinza da imagem digital com domínio $\mathbb{E}$, $b \in \mathbb{Z}^{\mathbb{B}}$. Considere $b \in \mathbb{Z}^{\mathbb{B}}$ a função estruturantes (vizinhança/kernel), conforme exemplo abaixo, com origem sendo o seu centro. Considere a origem sendo o seu centro de b, conforme trecho de código abaixo:

```
def algoritmo(f, b):
  [l, c] = f.shape
  [bl, bc] = b.shape
  g = f.copy()
  for xi in range(l):  # para cada linha xi de f
    for xj in range(c):  # para cada coluna xj de f
      for bi in range(bl):  # para cada linha bi de b
        for bj in range(bc):  # para cada coluna bj de b
          yi = int(xi + bi - bl / 2 + 0.5)  # ajustar vizinho (yi,yj) de (xi,xj), definido por b
          yj = int(xj + bj - bc / 2 + 0.5)  # considerando origem o centro de b
          # incluir aqui os cálculos de vizinhança
  return g
```

Implemente a sua solução para o **filtro morfológico** a seguir:

$$S_n(f,b) = (f \ominus nb) - (f \ominus nb) \circ b; \qquad esqueleto(f,b) = \bigcup_n S_n(f,b)$$

Sugestão: usar np.logical_* (ver `http://felix.abecassis.me/2011/09/opencv-morphological-skeleton`).

Submeter o arquivo **Q1.py** (com a resposta).

Exemplo (considerar somente os números como elementos de entrada/saída para os casos de teste):

```
b:
 0  0  0
 0  0  0
 0  0  0
f:
 1  1  1  0  0  0  0  0  0
 1  1  1  1  1  1  1  1  1
 1  1  1  1  1  1  1  1  1
 1  1  1  1  1  1  1  1  1
 1  1  1  1  1  1  1  1  1
 1  1  1  1  1  1  1  1  1
 1  1  1  1  1  1  1  1  1
 0  0  0  1  1  1  1  1  1
resultado:
 1  0  0  0  0  0  0  0  0
 1  0  0  0  0  0  0  0  0
 1  0  0  0  0  0  0  0  0
 1  1  1  1  1  0  0  0  0
 1  1  1  1  1  0  0  0  0
 0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  1
 0  0  0  0  0  0  0  0  1
```