†

Universidade Federal do ABC
Bacharelado em Ciência da Computação
Disciplina: Processamento Digital de Imagens
Prof.: Francisco Zampirolli
Turma: Imagem 2020 **Sala:** 123
Exame: Atividade 7 **Data:** 07-06-2020

Ass.: _____
Estudante: Marcelo Pena **ID/RA:** 11039314

#138 - 2020-05-27 - 09:21:04

Instruções:

(a) Esta é a Atividade 7 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 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 $\mathbb{E}$, com $b \in \mathbb{Z}^{\mathbb{B}}$ a função estruturantes (vizinhança/kernel). Considere a origem de b sendo o seu centro, conforme trecho de código abaixo:

```
def erosao(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
          if 0 <= yi < l and 0 <= yj < c:  # para não sair do domínio E da imagem f
            # incluir aqui os cálculos de vizinhança
  return g
```

Considere agora a seguinte função estruturante variável b_i, para $i = 0, 1, 2, \cdots$:

$-4i+2$	$-2i+1$	$-4i+2$
$-2i+1$	$\mathbf{0}$	$-2i+1$
$-4i+2$	$-2i+1$	$-4i+2$

Implemente o seguinte algoritmo da Transformada Distância Euclidiana (TDE), conforme exemplo a seguir:

$TDE : K^{\mathbb{E}} \times \mathbb{Z}^B \times \mathbb{Z}^B \times \cdots \longrightarrow K^{\mathbb{E}}$
$\quad TDE(f, b_1, b_2, \ldots) = g$

$g = f$;
$i = 0$;
$\forall g \neq \varepsilon_{b_i}(g)$
$\quad g = \varepsilon_{b_i}(g)$;
$\quad i++$;

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

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

```
f:
99 99 99 99 99 99 99 99 99
99 99 99 99 99  0 99 99 99
99 99 99 99 99 99 99 99 99
99 99 99 99 99 99  0 99  0
99 99 99 99 99 99 99 99 99
99 99 99 99 99 99 99 99 99
99  0 99 99  0 99 99 99 99
99 99 99 99 99 99 99 99 99
resultado:
26 17 10  5  2  1  2  5  9
25 16  9  4  1  0  1  4  4
17 16 10  5  2  1  1  2  1
10  9 10  8  4  1  0  1  0
 5  4  5  5  4  2  1  2  1
 2  1  2  2  1  2  4  5  4
 1  0  1  1  0  1  4  9  9
 2  1  2  2  1  2  5 10 16
iteracoes:
6
```