# Processamento de Imagens
# Tarefa 1

Vinicius Gasparini

24 de junho de 2020

## Parte I

### Questão 1

- Podemos citar como exemplo os formatos PNG, JPEG, BMP, GIF e TIF. Cada um desses formatos possuem suas individualidades e usos mais adequados.
- Uma imagem digital possui diversos tipos de dados associados, esses dados estão intrinsecamente ligados ao conteúdo da cena e como ela deverá ser exibida. De maneira sucinta, existem 4 grandes tipos de imagens e por consequência, cada uma com suas especifidades de dados:
    1. Imagens binárias: são imagens preto e branco da maneira mais pura, sem sub-tons. Se trata de uma matriz de ordem 2 que aloca um valor do conjunto {0,1}.
    2. Imagens em escala de cinza: uma evolução por assim dizer da anterior. Agora a matriz possui inteiros de $N$ bits, onde os extremos do valor representam o preto e branco e os valores intermediários representam os $2^N$ tons de cinza possíveis.
    3. Imagens RGB: são imagens representadas agora por matrizes de ordem 3, cada pixel portanto é descrito por uma tripla que caracteriza os canais vermelho, verde e azul. Podem existir variações deste formato como por exemplo com a adição de um canal que descreva o Alfa (transparência), padrão esse utilizado em imagens PNG.
    4. Imagens em ponto flutuante: este formato é utilizado quando a imagem não necessariamente representa valores de luz ou cor. Para isso são utilizados números com precisão definida de maneira a caracterizar a intensidade do dado.
- Para o script, foi utilizado o filtro luma $Y'$ conforme equação. A imagem utilizada neste exemplo é a *casa.png* fornecida no repositório da disciplina.

$$Y' = 0.299R + 0.587G + 0.114B$$

```
import numpy as np
import matplotlib.pyplot as plot
import matplotlib.image as mpimg

def rgb_to_gray(rgb):
    return np.dot(rgb[..., :3], [0.2989, 0.5870, 0.1140])

img = mpimg.imread("casa.png")
gray = rgb_to_gray(img)
plot.imshow(gray, cmap=plot.get_cmap("gray"), vmin=0, vmax=1)
plot.axis("off")
plot.savefig("casa_grayscale.png", bbox_inches="tight", pad_inches=0)
```

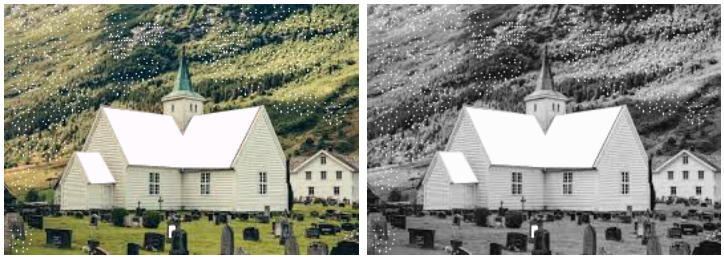

Figure 1: Imagem original na esquerda, modificada pelo $rgb_to_grayadireita$.

## Questão 2

- Grandes imagens influenciam diretamente a performance de algoritmos de processamento pois as imagens são representadas por matrizes $NxM$, deste modo, de maneira básica, os algoritmos possuem tempo de execução $O(NxM)$. Com $N \geq M$, temos $O(N^2)$. Dai a necessidade de se desenvolver *hardwares* dedicados ao processamento paralelo de imagens, as *GPU*.

## Questão 3

- Ruído é uma variação aleatória causada na obtenção ou processamento da imagem. Essa pertubação pode ter origem externa ou interna ao algoritmo de processamento, sendo seu controle difícil devido imprevisibilidade de ocorrência.
  Como as imagens, em sua maioria, não passam somente uma etapa de processamento, estes ruídos podem ser causados nas diversas etapas da obtenção da imagem.
  As principais causas de ruído por fatores externos estão ligadas ao momento de captura. Um ruído nessa etapa pode surgir por meio da vibração, saturação do sensor, subexposição, distorção de lente e até mesmo poeira no ambiente. Ruídos podem ocorrer também durante o processamento da informação pelo sensor. Como por exemplo erros de ponto flutuante, *overflow* e aproximações matemáticas de constantes, como é o caso do $\pi$.
  Pós registro pelo sensor, a imagem pode ainda ser acarretada por ruídos na etapa de codificação. Isto é, uma vez que uma imagem é adquirida pela lente, processada pelo sensor, ela é comprimida em formatos como o JPEG e PNG. Neste processo de compressão são gerados alguns artefatos provenientes do processo de simplificação da informação. Como solução, existem variações do próprio PNG que garante compressão sem perdas.

### Questão 4

- A subtração de *frames* de um vídeo resultará nos pixels diferentes entre os dois momentos. Com isso é possível perceber quais trechos da cena se mantiveram estáticos e quais se modificaram. As regiões em preto indicarão onde ocorreu o movimento/alteração de conteúdo.

### Questão 5

- O algoritmo abaixo é responsável por dado uma matriz em escala de cinza e um limiar, é modificado a matriz da imagem original para a sua formação binária filtrada.

```
def apply_threshold(img, threshold):
    h, w = len(img), len(img[0])
    for i in range(h):
        for j in range(w):
            if img[i][j] > threshold:
                img[i][j] = 1
            else:
                img[i][j] = 0
    return img
```

Figure 2: Imagem em tons de cinza na esquerda, modificada pelo algoritmo de limiar a direita. Foi aplicado limiar de *0.70*

# Parte II

### Questão 16

- A área do polígono $ABCD$ projetado é de 225.0 $mm^2$
- Segue abaixo lista dos pontos projetados
  - Ponto a' = (1.51, 15.0) mm
  - Ponto b' = (1.53, 15.0) mm
  - Ponto c' = (15.0, 1.55) mm
  - Ponto d' = (15.0, 1.51) mm
  - Ponto e' = (1.45, 1.53) mm
  - Ponto f' = (1.44, 1.53) mm
- Não, as área cinzas foram totalmente modificadas. Os pontos são muito próximos, dando a impressão de não haver mais separações entre os diferentes objetos.
- O algoritmo utilizado para o calculo da projeção está ao fim dessa seção.

## Questão 17

- A tecnologia *Complementary metal–oxide–semiconductor* (CMOS) processa a imagem pelo método *rolling shutter*, isto é, a captura da luz ocorre linha a linha pelos receptores. Já a tecnologia *Charge-coupled device* (CCD) registra a imagem através do método *global shutter*, capta a luz da cena de uma só vez. Cada método possui suas vantagens, CMOS se destaca no registro de imagens em maior resolução e qualidade de cor, portanto mais favorável a captura de pessoas caminhando no um corredor de um aeroporto. Deste modo, CCD é mais recomendado ao registro de cenas mais movimentadas pela maior velocidade de registro, indicada a capturar o helicóptero.

## Questão 18

- Para solucionar este problema é possível utilizar apenas 3 sensores simples. Um para identificar a cor vermelha, um para a cor verde e outro para a cor azul. Caso os três sensores captarem simultaneamente, isso significa que o carro é da cor branca. Isso se deve ao fato do padrão RGB mensurar a quantidade de *red, green e blue*. Como é um padrão de cor aditiva, a soma das três variáveis resulta no branco.

```python
import numpy as np

def matriz_foco(d):
    m = np.zeros((3, 4))
    for i in range(3):
        for j in range(4):
            if i == j:
                m[i][j] = 1
    m[2][2] = 1 / d
    return m

def gera_ponto(x, y, z):
    return np.matrix([[x], [y], [z], [1]])

def multiplica(matriz, ponto):
    return matriz * ponto

def escala(n, escala=2):
    return float(n) / 10 ** escala

def homogeneo_cartesiano(ponto):
    ponto[0][0] /= ponto[2][0]
    ponto[1][0] /= ponto[2][0]
    return ponto

matriz_perspectiva = matriz_foco(5)

a = np.matrix([150.75, 1500, 500, 1]).transpose()
b = np.matrix([153.5, 1500, 500, 1]).transpose()
c = np.matrix([1500, 155, 500, 1]).transpose()
d = np.matrix([1500, 150.75, 500, 1]).transpose()
e = np.matrix([145, 153, 500, 1]).transpose()
f = np.matrix([144.3, 153, 500, 1]).transpose()

pontos,nomes = [a, b, c, d, e, f],["a'", "b'", "c'", "d'", "e'", "f'"]
A = np.matrix([0, 1500, 505, 1]).transpose()
B = np.matrix([1500, 1500, 505, 1]).transpose()
C = np.matrix([0, 0, 505, 1]).transpose()
D = np.matrix([1500, 0, 505, 1]).transpose()

projecoes = [A, B, C, D]
for p in range(len(projecoes)):
    projecoes[p] = multiplica(matriz_perspectiva, projecoes[p])
seg_AC = projecoes[0][1] - projecoes[2][1]
seg_CD = projecoes[3][0] - projecoes[2][0]
area = seg_AC * seg_CD

print(f"Area = {escala(area,4)} mm2\n")
for nome, ponto in zip(nomes, pontos):
    print(
        "Ponto {} = [{:.3}, {:.3}, {:.3}]".format(
            nome, *map(escala, multiplica(matriz_perspectiva, ponto))
        ))
```