

# DIM0451 - Tópicos Especiais em Computação XII

## Processamento Digital de Imagens e Visão Computacional

Gabriel Araújo de Souza  
Marcus Paulo Soares Dantas  
Rafael Garcia Dantas

Listas de Exercícios

24 de fevereiro de 2019

### Questão 01.

As células fotorreceptoras podem ser divididas em dois grupos: bastonetes e cones. Os bastonetes são sensíveis a baixa iluminação, sendo assim responsáveis pela visão noturna. Os cones são sensíveis a alta iluminação e fazem com que tenhamos a percepção das cores. Este último possui uma concentração maior em uma região denominada fóvea, já os bastonetes se concentram nas extremidades da retina (Sem concentração na fóvea central e mácula).

### Questão 02.

Existem três tipos de cones (S, M e L). Os cones do tipo S são sensíveis a baixas frequências e portanto reconhecem a cor azul. Os cones M reconhecem uma frequência de nível médio e associam a cor verde. Os do tipo L são sensíveis a altas frequências e reconhece a cor vermelho. Quando uma luz amarela é projetada na retina, dois tipos de cones ativados, o cone L (vermelho) e o cone M (verde), desse forma temos a sensação da cor amarelo.

### Questão 03 (A).

```
# load image
src = cv2.imread("img/img1.jpg")

# separate channels
b,g,r = cv2.split(src)

# load
src_h, src_w, _ = src.shape

# create matrix of zeros
zeros = np.zeros((src_h, src_w), dtype= 'uint8')

# merge channels
img_blue = cv2.merge((b, zeros, zeros))
img_green = cv2.merge((zeros, g, zeros))
img_red = cv2.merge((zeros, zeros, r))

# plot images
cv2.imwrite("img/blue.png", img_blue) # blue channel
```

```
cv2.imwrite("img/green.png", img_green) # green channel  
cv2.imwrite("img/red.png", img_red) # red channel
```



1.1 Imagem Original



1.2 Canal Vermelho (Red)



1.3 Canal Verde (Green)



1.4 Canal Azul (Blue)

### Questão 03 (B).

```
image = cv2.imread("img/img1.jpg")  
horizontal_image = cv2.flip( image, 0)  
cv2.imwrite("img/img1_horizontal.png", horizontal_image)
```



1.5 Imagem Original



1.6 Imagem Rotacionada

### Questão 03 (C).

```
peixe1 = cv2.imread("img/peixe1.jpg")  
peixe2 = cv2.imread("img/peixe2.jpeg")  
alpha = 0.5  
beta = (1.0 - alpha)  
dst = cv2.addWeighted(peixe1, alpha, peixe2, beta, 0.0)  
cv2.imwrite("img/peixeblanding.png", dst)
```



1.7 Imagem 1



1.8 Imagem 2



1.9 Blending

### Questão 03 D.

```
# create matrix of zeros
value = 0
cont = 0
gradient = np.zeros((510,510,3), dtype= 'uint8')
for i in range (510):
    for j in range (510):
        gradient[i,j] = [value, value, value]
        if value < 255 and cont == 2:
            value = value +1
            cont = 0
        cont += 1
cv2.imwrite("img/gradient.png", gradient)
```



Figura 1: Gradient

### Questão 04.

- (A) Os números P1 e P4 representam o tipo Portable BitMap, o primeiro para a tabela ASCII e o segundo para o tipo binário. Os formatos ASCII são legíveis para o humano e facilitam a transferência para outras plataformas. Os formatos binários são mais eficientes no tamanho do arquivo, mas podem ter problemas de

ordem de bytes. Esses representam uma imagem preto e branco onde o preto é representado pelo valor 1 e o branco por 0. Ambos representam imagens com a extensão .pbm.

P2 e P5 são usados para o tipo Portable GrayMap com a extensão .pgm. O P2 informa que a representação será pela tabela ASCII e o P5 do tipo binário. Os pixels tem valores de 0-255 em escala de cinza.

Para o P3 e o P6 com extensão .ppm (Portable PixMap), a representação é colorida com intensidades de 0-255 para cada cor (R de Red, G de green e B de blue). A representação segue o mesmo modelos dos demais formatos

**(B)** Resultado após conversão



2.1 Formato JPG



2.2 Formato PBM

**(C)** Observe a tabela para vizualizar os resultados das conversões

| Formato | Tipo    | Extensão | Tamanho  |
|---------|---------|----------|----------|
| JPG     | -       | .jpg     | 116,1 KB |
| PBM     | ASCII   | .pbm     | 266,5 KB |
| PBM     | Binário | .pbm     | 16,9 KB  |
| PPM     | ASCII   | .ppm     | 1,4 MB   |
| PPM     | Binário | .ppm     | 399,1 KB |

**(D)** Nos formato PBM, quando se usa o tipo ASCII, cada pixel é representado por um caractere da tabela ASCII. No entanto, quando o padrão binário é usado, cada pixel é representado por apenas um bit. Por isso, o formato binário do PBM ocupa menos espaço que o ASCII.

**(E)** O formato PPM binário utiliza 24 bits por pixel (8 bits para cada tipo do RGB), enquanto que o PBM binário utiliza apenas 1 bit por pixel. Sendo assim, o formato binário do PPM será maior que o formato binário do PBM.

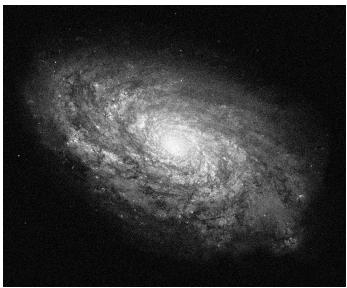
**Questão 05.**

Os formatos SVG e EPS são formatos vetoriais, já os demais são do formato bitmap.

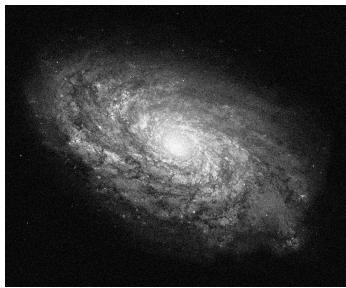
**Questão 06.**

```
# define path of paste
paste = 'img/ruido'
# read all images
paths = [ os.path.join(paste, name) for name in os.listdir(paste) ]
```

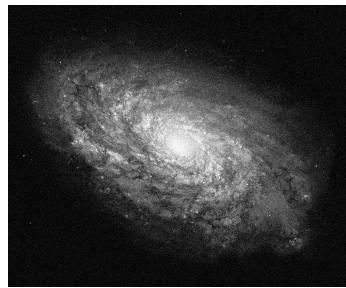
```
images = [ cv2.imread(name) for name in paths ]
# read images dimensions
h, w, _ = images[0].shape
size = len(images)
# create a new image
average = np.zeros((h,w,3), np.float)
# average pixels
for i in range (h):
    for j in range (w):
        soma = [0,0,0]
        for k in range (size):
            soma += images[k][i,j]
        average[i,j] = soma/size
# Round values in array and cast as 8-bit integer
average = np.array(np.round(average),dtype=np.uint8)
# print image
cv2.imwrite("img/average.png", average)
```



2.3 Imagem 1



2.4 Imagem 2



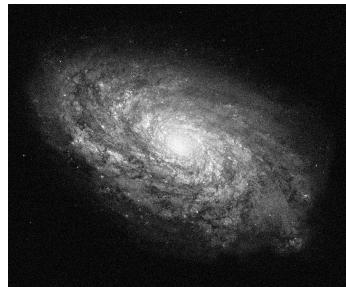
2.5 Imagem 3



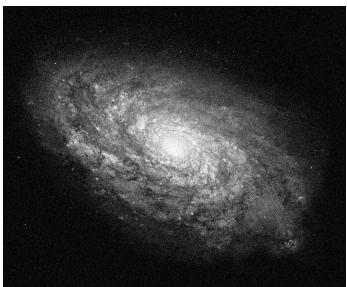
2.6 Imagem 4



2.7 Imagem 5



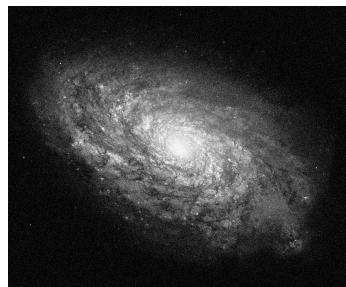
2.8 Imagem 6



2.9 Imagem 7



2.10 Imagem 8



2.11 Imagem 9



2.12 Média