

Nome do arquivo a ser entregue: **p14.py**

Faça o *download* do arquivo **imagens.pyc**. Em seguida, faça o download do arquivo **p14.py**, entre no IDLE e abra esse arquivo dentro do IDLE. Execute o programa e veja se está funcionando corretamente. Deve aparecer na tela as figuras mostradas abaixo:



A primeira imagem é a original. A outra é a imagem com a aplicação de um filtro, onde removemos todo o vermelho, deixando apenas o verde e o azul.

Vamos dar uma olhada no código do programa e nas funções já implementadas:

```

1  # Nome do aluno:
2  # Matrícula:
3  # Data:
4  # Este programa aplica uma série de filtros em uma imagem, modificando a cor
5  # dos pixels.
6
7  import imagens
8
9  def aplicaFiltro( imagem, filtro ):
10     im2 = imagem.copia()
11     m = im2.altura
12     n = im2.largura
13     for i in range( 0, m ):
14         for j in range( 0, n ):
15             r, g, b = im2[i][j]
16             r, g, b = filtro( r, g, b )
17             im2[i][j] = (r, g, b)
18     im2.mostrar()
19
20 # Esta função retorna a "luminância" correspondente aos componentes r,g,b
21 def luminancia( r, g, b ):
22     return int( 0.299 * r + 0.587 * g + 0.114 * b )
23
24 # retornar apenas as componentes verde e azul, zerando a vermelha
25 def filtroGB( r, g, b ):
26     return 0, g, b
27
28 im = imagens.Imagem('jardim.jpg')
29 im.mostrar()
30
31 aplicaFiltro( im, filtroGB )

```

Função `aplicaFiltro`

Essa função recebe dois parâmetros. O primeiro (*imagem*) é uma imagem colorida representada por uma matriz de pixels, conforme já visto no Exercício 2 da Aula 08 e na prática P12. O segundo parâmetro (*filtro*) é uma função que recebe as três componentes de cor (*r*, *g*, *b*) e retorna essas componentes alteradas de alguma forma.

Na linha 10 é feita uma cópia da imagem para dentro da variável `im2`. As linhas 11 e 12 pegam o número de linhas e colunas da imagem, respectivamente. Nas linhas 13 e 14 percorre-se toda a imagem. Cada pixel é desmembrado em suas componentes (linha 15). Depois, a função `filtro` é chamada, passando essas componentes, e pegando de volta as cores transformadas (linha 16). Por fim, o pixel da imagem é alterado com essas novas cores (linha 17). Na linha 18, a nova imagem é mostrada na tela.

Função `luminancia`

Essa função recebe como parâmetros as três componentes (r, g, b) de um pixel e retorna a “luminância” ou “brilho” desse pixel. Você precisará dessa função para implementar dois dos filtros propostos.

Função `filtroGB`

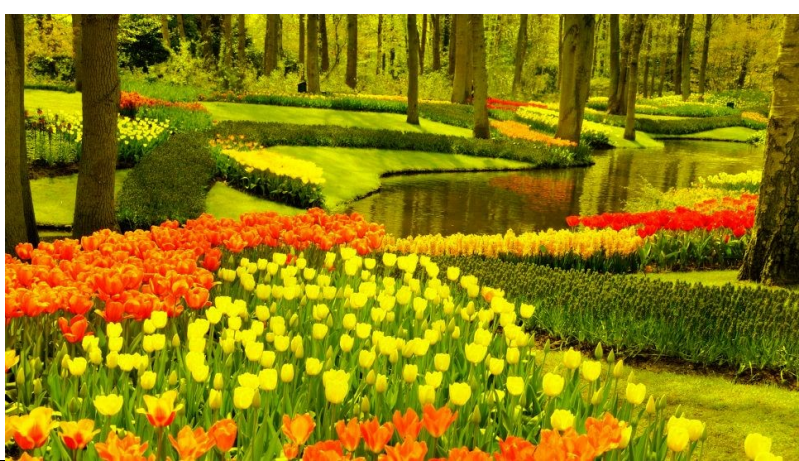

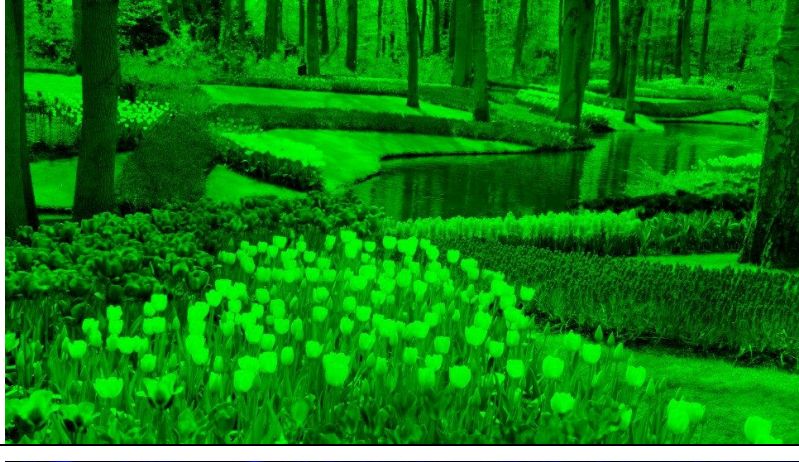

Essa função recebe como parâmetros as três componentes (r, g, b) de um pixel e retorna apenas as componentes verde e azul, “zerando” a componente vermelha.


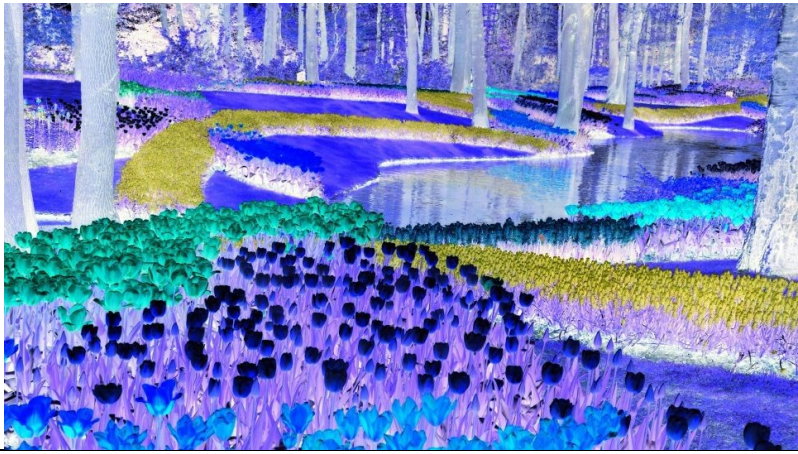


Corpo principal do programa

A linha 28 lê uma imagem do disco e cria a variável `im` contendo essa imagem. A linha 29 mostra a imagem na tela. A linha 31 chama a função `aplicaFiltro` passando como parâmetros a imagem `im` e a função `filtroGB`. Isso fará com que seja criada e mostrada na tela uma nova imagem com as cores alteradas através da aplicação desse filtro.

Nesta prática, você deverá completar esse programa para que, além das duas imagens já mostradas, ele mostre na tela as outras imagens descritas a seguir. Para isso, você deverá implementar a função de filtro apropriada para cada caso, e depois chamar a função `aplicaFiltro`, passando como parâmetros a imagem `im` e a função implementada, da mesma forma como foi feito no exemplo descrito acima.

| Descrição | Resultado |
|--|--|
| Produzir uma imagem apenas com as componentes vermelha e azul. |  |

| | |
|--|--|
| <p>Produzir uma imagem apenas com as componentes vermelha e verde.</p> |  |
| <p>Produzir uma imagem apenas com a componente vermelha.</p> |  |
| <p>Produzir uma imagem apenas com a componente verde.</p> |  |
| <p>Produzir uma imagem apenas com a componente azul.</p> |  |

| | |
|---|--|
| <p>Produzir uma imagem trocando a componente vermelha pela verde, a verde pela azul, e a azul pela vermelha. Ou seja, trocar a sequência r, g, b pela sequência g, b, r.</p> |  |
| <p>Produzir a imagem “negativa”. Para isso, cada componente x deve ser substituída por seu complemento, $255-x$.</p> |  |
| <p>Produzir uma imagem em tons de cinza. Para isso, deve-se obter a “luminância” ou “brilho” do pixel e substituir cada uma das três componentes por essa luminância.</p> |  |
| <p>Efetuar a “binarização da imagem por limiar simples”. Para isso, faça o seguinte:</p> <ol style="list-style-type: none"> 1. Obtenha o brilho B do pixel; 2. Se B for menor que um determinado limiar, retorne a cor preta (0,0,0). Caso contrário, retorne branco (255,255,255). <p>O resultado disso será uma imagem binária preto & branco. Use um limiar = 128.</p> |  |
| <p>Fazer uma transformada <i>gamma</i> na imagem</p> | |

Ao capturar uma imagem usando um dispositivo (e.g. uma câmera digital) e exibir essa imagem em outro dispositivo (e.g. uma TV LCD), é possível que a imagem fique com uma aparência um pouco diferente, em geral um pouco mais clara ou mais escura. Essa distorção pode ser corrigida com *correção gamma*.

Uma *correção gamma* em um pixel é uma transformação do tipo: $r = s^\gamma$

onde s é o valor original do pixel. Valores de γ maiores que 1 escurecem a imagem, e valores menores que 1 clareiam a imagem. Use o valor $\gamma = 0,7$ e aplique essa transformada para cada uma das três componentes (r, g, b) da imagem.

Só um detalhe: para essa operação, os valores de r e s devem estar normalizados, ou seja, sempre no intervalo [0,1]. Para obter um valor de pixel nesse intervalo, basta dividi-lo por 255. Para retornar o valor normalizado para o valor correto a ser exibido, basta multiplica-lo por 255.

Por exemplo, para a componente r, teremos:

$$r = \text{int}((r/255) ** \text{gamma} * 255)$$

Se você colocar a imagem resultante ao lado da original, verá que ela ficou um pouco mais clara. Note como os troncos das árvores ficaram mais nítidos.



Produzir uma imagem em tom de "sépia"

Para obter essa imagem, devemos aplicar a seguinte transformada nos componentes:

$$\begin{bmatrix} r_s \\ g_s \\ b_s \end{bmatrix} = \begin{bmatrix} 0,393 & 0,769 & 0,189 \\ 0,349 & 0,686 & 0,168 \\ 0,272 & 0,534 & 0,131 \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix}$$

Onde (r, g, b) representa o pixel original e (r_s, g_s, b_s) representa o pixel transformado.

Obs.: assim como foi mostrado na *correção gamma* acima, as três componentes devem ser retornadas como valores inteiros. Por exemplo:

$$rs = \text{int}(r * 0.393 + g * 0.769 + b * 0.189)$$



Quantização da imagem para n níveis em cada componente

Em geral são usados 8 bits (ou 1 byte) para representar cada componente de cor. Isso dá $2^8 = 256$ níveis para cada componente, totalizando $256^3 \approx 16$ milhões de cores para uma imagem RGB.

Mas podem haver ocasiões em que seja necessário usar uma quantidade maior de níveis por componente (e.g. em imagens médicas) ou um número menor de níveis (e.g. para reduzir o tamanho em bytes das imagens para armazenamento e transmissão).

Para transformar uma imagem RGB “tradicional” de 8 bits para n níveis, devemos aplicar a seguinte fórmula:

$$\begin{aligned} \text{fatorConversao} &= 255 / (n-1) \\ x2 &= \text{round}(\text{int}(x/\text{fatorConversao} + 0.5) * \text{fatorConversao}) \end{aligned}$$

onde x representa cada componente original, e $x2$ é a componente transformada. Por exemplo, para $n = 4$ serão necessários apenas 2 bits para cada componente (uma redução de 4x no tamanho da imagem em bits). Use esse valor para obter a imagem mostrada abaixo.

$$\begin{aligned} n &= 4 \\ \text{fatorConversao} &= 255 / (n-1) \\ r2 &= \text{round}(\text{int}(r/\text{fatorConversao} + 0.5) * \text{fatorConversao}) \\ g2 &= \text{round}(\text{int}(g/\text{fatorConversao} + 0.5) * \text{fatorConversao}) \\ b2 &= \text{round}(\text{int}(b/\text{fatorConversao} + 0.5) * \text{fatorConversao}) \end{aligned}$$



Testando outra imagem

Troque o nome do arquivo no início do programa. No lugar de **jardim.jpg**, use o **lenna.jpg** e execute o programa novamente. **Obs.:** Você pode entregar o programa usando qualquer uma dessas duas imagens.

☞ Não esqueça de preencher o cabeçalho com seus dados e uma breve descrição do programa.

Após certificar-se que seu programa está correto, envie o arquivo do programa fonte (**p14.py**) através do sistema do LBI.

Brincando um pouco mais

Depois de entregar seu trabalho, se sobrar tempo, brinque um pouco mais com as transformadas acima:

1. Na binarização, altere o valor do limiar para que fique mais adequado para a Lenna.
2. Na transformada gamma, faça testes com o valor de gamma em 0,5 e 1,5, por exemplo.