



Universidade Federal
do Espírito Santo

Departamento de Computação e Eletrônica - CEUNES

ESTRUTURA DE DADOS I

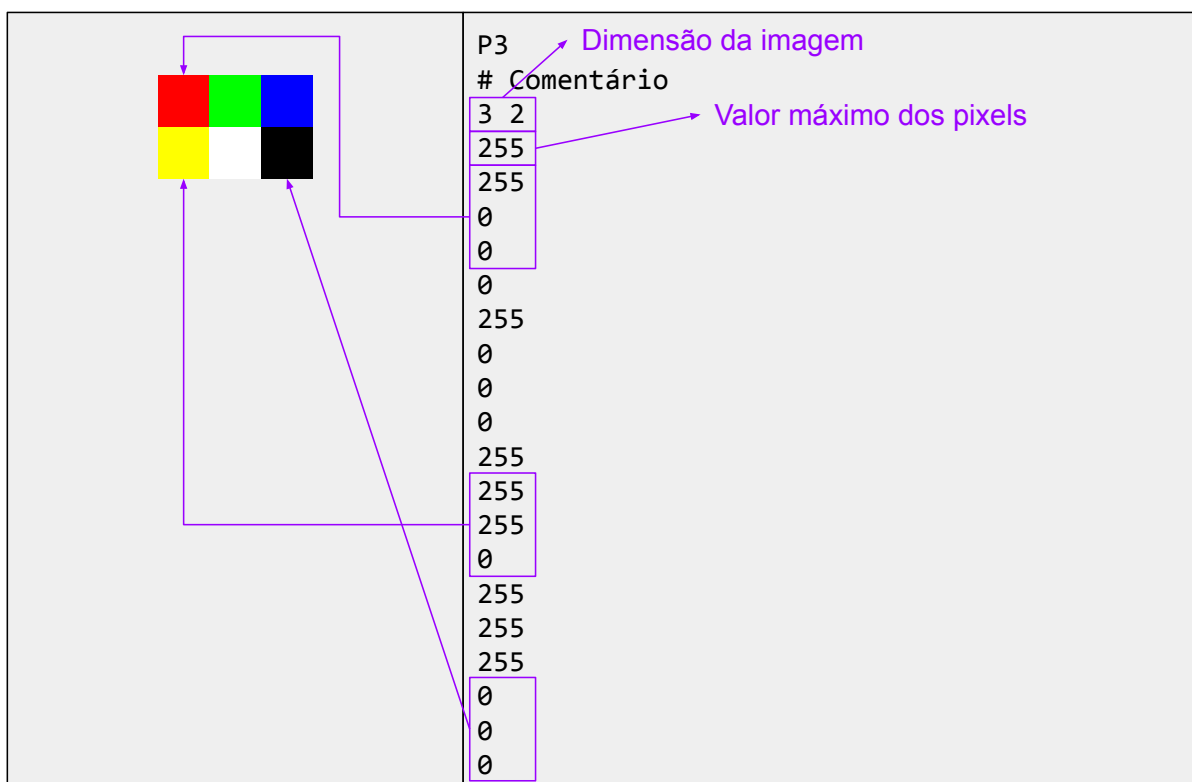
Prof. Oberlan Romão

Exercício Programa 1

Dicas Gerais

Formato PPM

O arquivo PPM é escrito em formato texto, você pode abri-lo em qualquer editor de texto simples, como VS Code e até mesmo modificá-lo. Quando aberto em um editor de imagens, como o GIMP, ele é interpretado como uma figura, e os valores indicam a cor de cada pixel, como mostrado no exemplo abaixo.



Dicas gerais

Para esse EP, sugiro que sigam os passos abaixo, fazendo um de cada vez, e só passando para o próximo quando o anterior funcionar perfeitamente (sempre teste vazamento de memória antes de prosseguir).

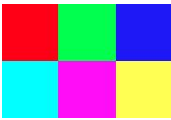

- `alocaImagem (Imagem.c)`: A função deve alocar um TAD Imagem e retornar um ponteiro para a área de memória alocada. Essa função deve usar a função `mallocSafe`;
- `liberaImagem (Imagem.c)`: Faz a desalocação de memória de um ponteiro do TAD Imagem;
- `salvaImagem (Imagem.c)`: Salva a imagem em um arquivo com nome `nomeArquivo`. Após a implementação dessa função, teste o seu programa para ver se tudo está saindo como esperado. Abra programa e use a opção de salvar para gerar uma cópia do arquivo. Com exceção do comentário, todo o restante dos dados devem ser exatamente iguais;
- `escurecerImagem, clarearImagem, escalaDeCinzaImagem (Filtros.c)`: Esses filtros, explicados abaixo, são os mais simples de serem implementados. Comecem por eles. Note que um `Pixel` é um `struc` que contém um array de tamanho 3 do tipo `Byte` (`unsigned char`), faça a conversão sempre que precisar fazer contas;
- `copiarImagem (Imagem.c)`: A função recebe um ponteiro para o TAD Imagem e retorna uma cópia. Use a opção de desfazer ou voltar a imagem original para testar a função;
- `deteccaoBordasLaplace, filtroSobel (Filtros.c)`: os dois filtros são mais complexos, deixe-os para o fim, assim você já estará mais familiarizado com o TAD;
- `meuFiltro (Filtros.c)`: esse é por sua conta! Pense em todos os filtros/efeitos que fez e implemente o seu filtro. Se quiser, pode pesquisar por “matriz de convolução” ou filtros de imagens. Seja criativo.

Cuidado ao efetuar contas com os valores `unsigned char`! Qualquer valor acima de 255 ou abaixo de 0 será convertido em algo dentro dessa faixa. Por exemplo $255 + 1 = 256$, será guardado como 0. Da mesma forma $0 - 1 = 255$ e $255 + 10 = 9$. Como não é isso que você quer, ao efetuar qualquer conta, use uma variável do tipo `int`. No final, antes de converter de volta para `unsigned char`, considere qualquer valor acima de 255 como 255, e qualquer valor abaixo de 0 como 0.

Opções do programa

• Opção ‘1’: Escurecer imagem

Ao teclarmos ‘1’ na janela da imagem, será solicitado, no terminal, o fator de escurecimento da imagem. Para escurecer a imagem, reduzimos um certo valor de cada banda de cor de cada pixel da imagem. Note que valores menores que 0, permanecem como 0.

Imagem original	Imagem escurecida por fator 50
P3 # Comentário 3 2 255 255 0 0 0 255 0 0 0 255 0 255 255 255 0 255 255 255 0 	P3 # Comentário 3 2 255 205 0 0 0 205 0 0 0 205 0 205 205 205 0 205 205 205 0 

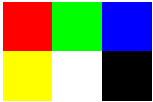

Esse fica por minha conta. Me devem um café :)

• Opção ‘2’: Clarear imagem

Será solicitado, no terminal, o fator de clareamento da imagem. Para clarear a imagem, aumentamos um certo valor de cada banda de cor de cada pixel da imagem. Note que valores maiores que 255, permanecem como 255.

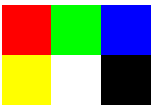

• Opção ‘3’: Imagem em escala de cinza

Para transformar uma imagem colorida em escala de cinza, podemos, para cada pixel, calcular a média dos valores de cada banda de cor e atribuir esse valor a cada componente de cor do pixel. Por exemplo, em um pixel composto por [255, 170, 30], cada banda de cor receberá o valor $\lceil \frac{(255+170+30)}{3} \rceil = 152$, ou seja, o mesmo se transforma em [152, 152, 152].

Imagem original	Escala de Cinza
P3 # Comentário 3 2 255 255 0 0 0 255 0 0 0 255 255 255 0 255 255 255 0 0 0 	P3 # Comentário 3 2 255 85 85 85 85 85 85 85 85 85 170 170 170 255 255 255 0 0 0 

Note que usar a média de cada componente não gera um bom resultado para as cores primárias, já que todas acabam ficando com o mesmo tom de cinza. Outra alternativa, muito utilizada, é que cada banca

de cor receba $[0.3 \times RED + 0.59 \times GREEN + 0.11 \times BLUE]$, por exemplo, em um pixel composto por $[255, 170, 30]$, cada componente de cor receberá $[0.3 \times 255 + 0.59 \times 170 + 0.11 \times 30] = 181$.

Imagem original	Escala de Cinza
P3 # Comentário 3 2 255 255 0 0 0 255 0 0 0 255 255 255 0 255 255 255 0 0 0 	P3 # Comentário 3 2 255 77 77 77 151 151 151 29 29 29 227 227 227 255 255 255 0 0 0 

Na implementação dessa operação, você pode escolher uma das opções ou, se quiser, pesquisar e propor outra alternativa.

• Opção ‘4’: Filtro de Sobel

O filtro Sobel pode ser usado para detecção de bordas em imagens digitais e consiste na aplicação de duas matrizes (chamadas de [matrizes de convolução](#)), que detectam os contornos na vertical e na horizontal. No EP você deve implementar uma versão simplificada desse filtro, mas que apresenta um excelente resultado. As matrizes utilizadas são

$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad \text{e} \quad G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

Essas matrizes são aplicadas a cada banda de cor de cada pixel da imagem. Os coeficientes na matriz indicam o peso de cada pixel vizinho. No exemplo abaixo, mostramos à esquerda uma parte da matriz de pixels da imagem original. À direita mostramos como a matriz é aplicada a 3 desses pixels.

...	
...	10	15	80	85	95	...	
...	15	40	120	130	131	...	$10*1 + 15*2 + 80*1 + 16*(-1) + 41*(-2) + 100*(-1) = -78$
...	16	41	100	120	120	...	$40*1 + 120*2 + 130*1 + 43*(-1) + 80*(-2) + 110*(-1) = 97$
...	17	43	80	110	130	...	$41*1 + 100*2 + 120*1 + 50*(-1) + 70*(-2) + 42*(-1) = 129$
...	19	50	70	42	20	...	
...	

Para cada pixel, são utilizados seus 8 pixels vizinhos. No caso da matriz os da linha de cima são somados (pesos 1 e 2), os da linha de baixo são subtraídos (pesos -1 e -2), e os da mesma linha não são considerados (peso 0). Abaixo o resultado para esses 3 pixels. Os demais devem ser calculados da mesma forma.

...
...						...
...		0				...
...			97			...
...			129			...
...						...
...

Note que valores abaixo de 0 devem ser fixados em zero e valores acima de 255 deve ser fixado em 255. Se o valor resultante estiver entre 0 e 255, esse é o novo valor. Note que o cálculo de cada pixel é independente do outro, ou seja, sempre se usa o valor **original**, e não o modificado.

Aplicando essa matriz a todos os pixels, a imagem resultante dará ênfase as bordas dos objetos da imagem (onde há mudanças brusca de cor). Da mesma forma, a aplicação da matriz fará a detecção de contornos, porém em outro sentido.

Você deve aplicar as duas matrizes e combinar os resultados. No EP isso pode ser feito de qualquer forma simples, como somar, tirar a média, manter o maior ou o menor valor, etc. Se quiser comparar o resultado, o GIMP já possui uma implementação do filtro Sobel.

- **Opção ‘5’: Detecção de bordas de Laplace**

A implementação do filtro de detecção de bordas de Laplace segue a mesma ideia do filtro de Sobel, mas agora existe apenas uma matriz de convolução (apresentada abaixo) que deve ser aplicada em cada banda de cor de cada pixel.

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

- **Opção ‘m’: Aplicar o filtro personalizado**

Na função `meuFiltro (Filtros.c)`, você deve pensar e aplicar algum filtro/efeito de sua preferência na imagem. A escolha é livre (seja criativo), mas você deve explicar (e mostrar exemplos) no relatório o filtro implementado.

- **Opção ‘o’: Voltar para a imagem original**

Se você implementou a função `copiarImagem (Imagem.c)` de forma correta, essa opção já deve funcionar.

- **Opção ‘z’: Desfazer a última modificação**

Se você implementou a função `copiarImagem (Imagem.c)` de forma correta, essa opção já deve funcionar.

- **Opção ‘s’: Salvar a imagem atual**

Se você implementou a função `salvarImagem (Imagem.c)` de forma correta, essa opção já deve funcionar.

- **Opção ‘x’: Sair**

Se você implementou a função `liberarImagem (Imagem.c)` de forma correta, essa opção encerra o programa fazendo toda a desalocação necessária.

Usando o GIMP para criar arquivos PPM

Para criar arquivos PPM você pode escolher qualquer imagem na internet e usar o GIMP para exportar a imagem no formato PPM. Após abrir a imagem escolhida no GIMP, para exportá-la no formato desejado, vá em “Arquivo -> Exportar como”. Na janela de Exportar imagem, escolha o formato ppm (mude a extensão no nome do arquivo também). Ao pedir para exportar, outra janela irá se abrir, escolha ASCII e clique em Exportar. Após exportar, abra o arquivo em um editor de texto para ver se tudo deu certo.

Alguns resultados esperados

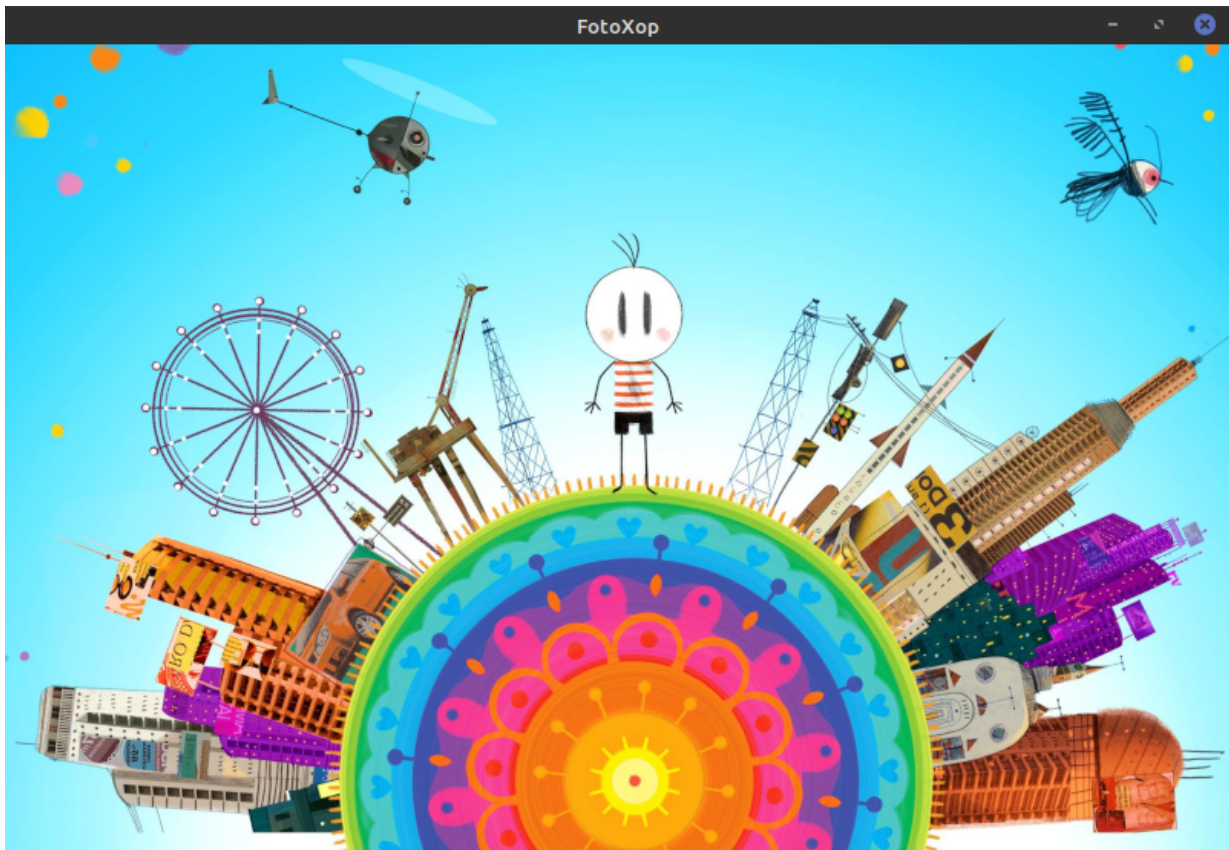


Figura 1: Original

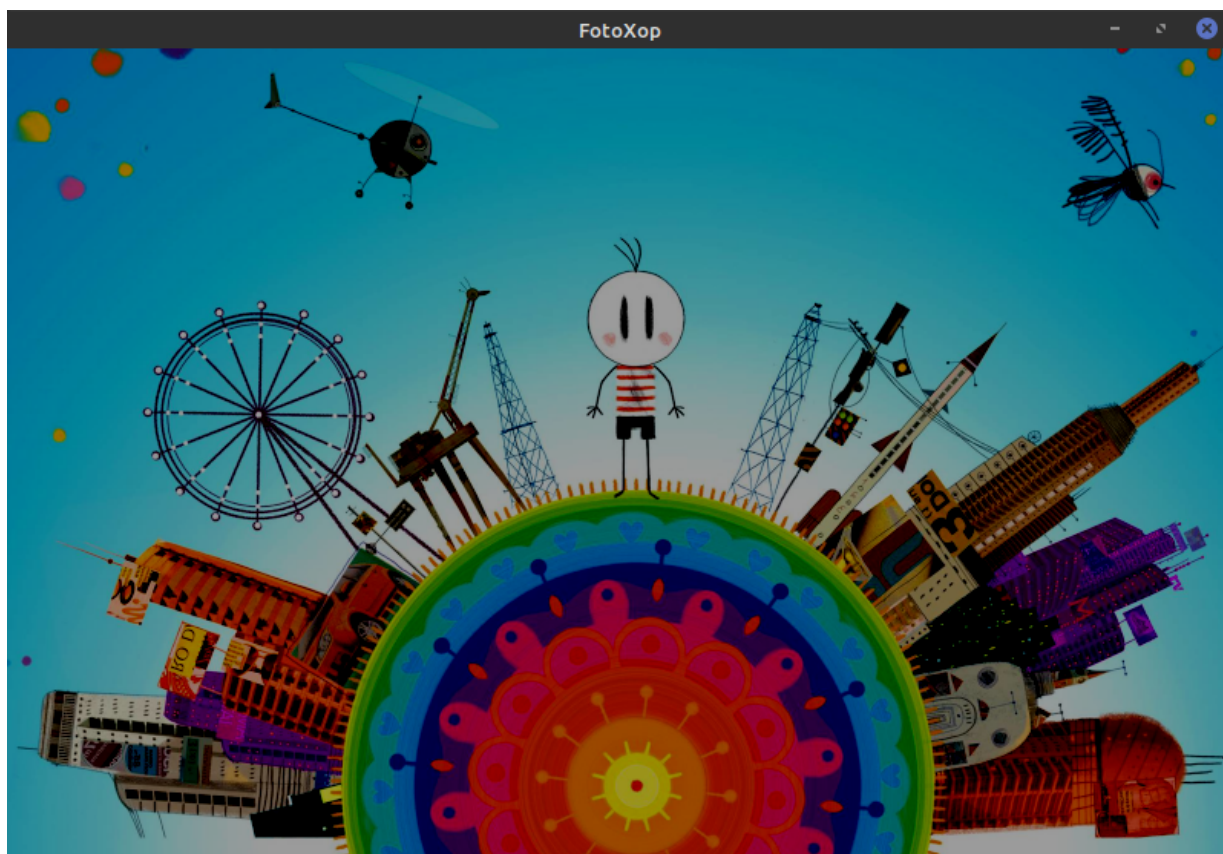


Figura 2: Escurecida - Fator = 100

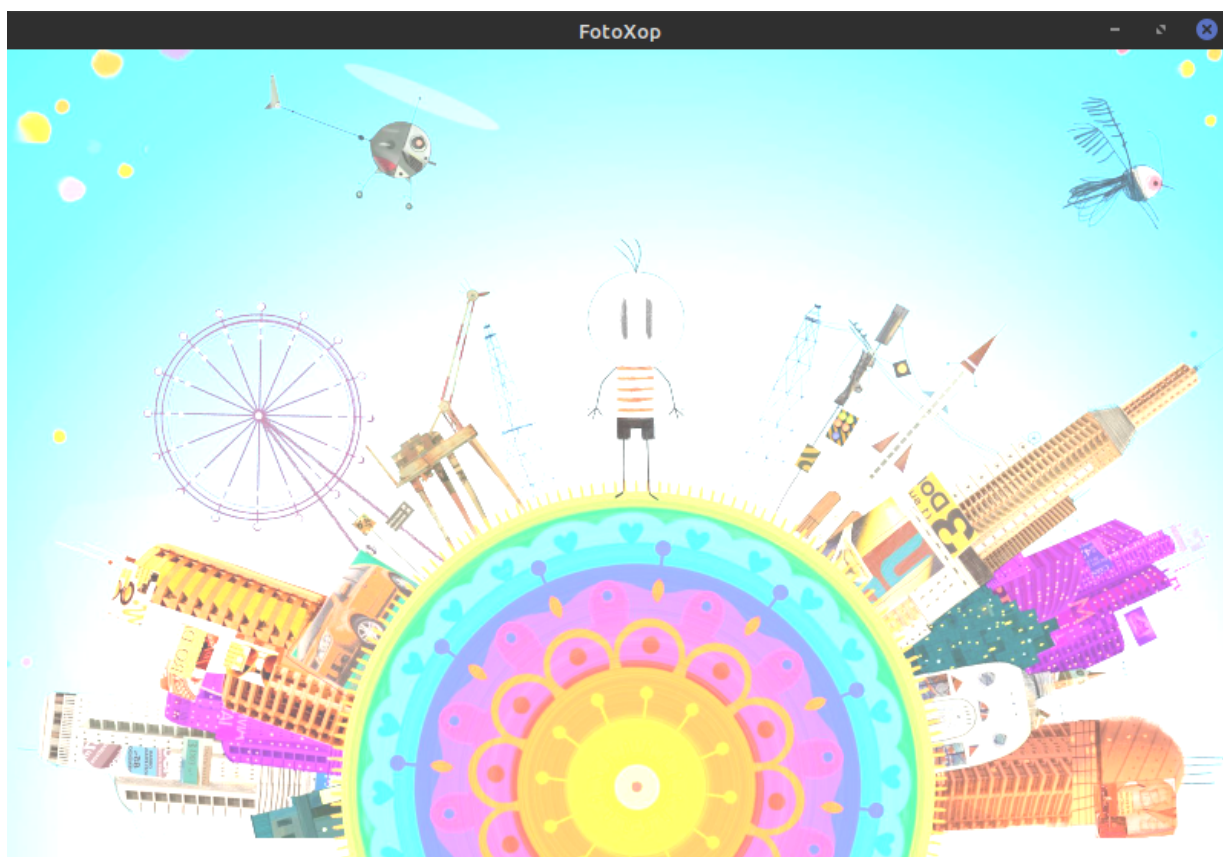


Figura 3: Clareada - Fator = 100

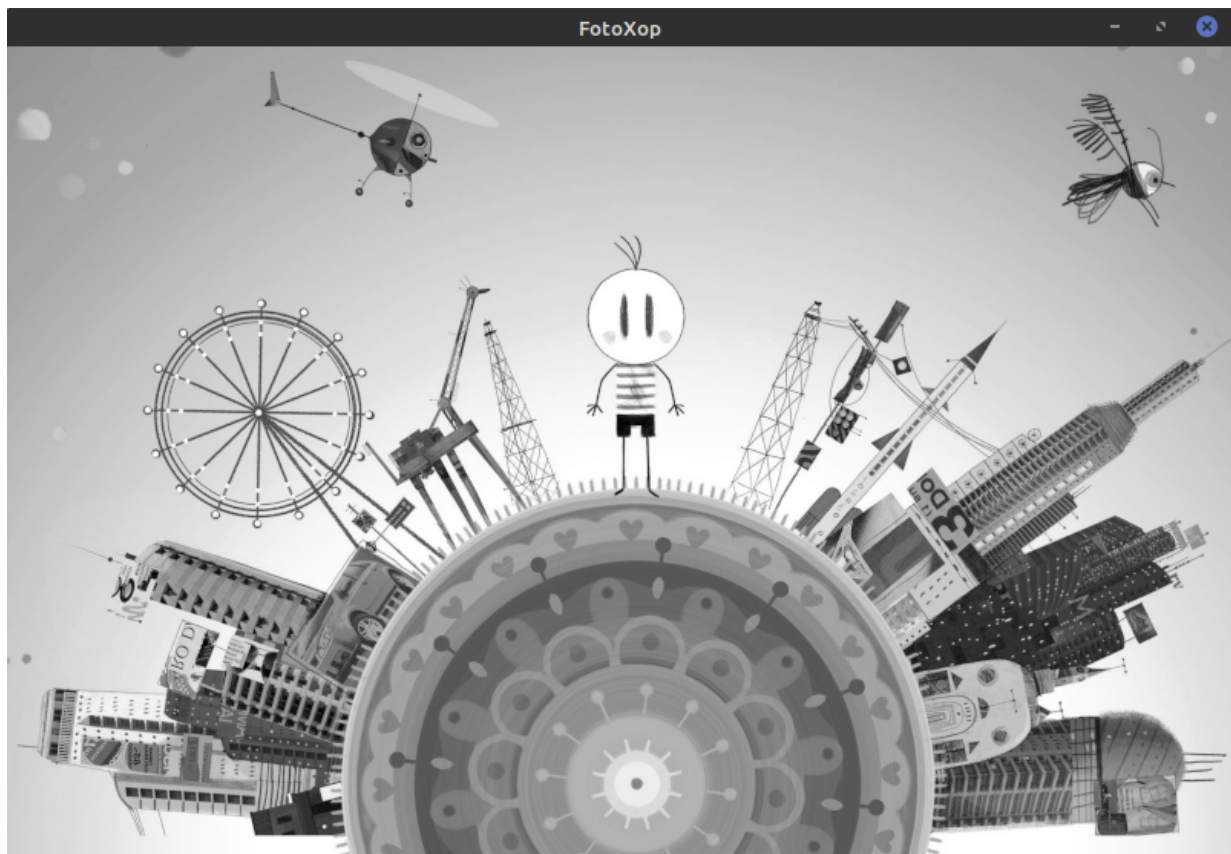


Figura 4: Escala de Cinza

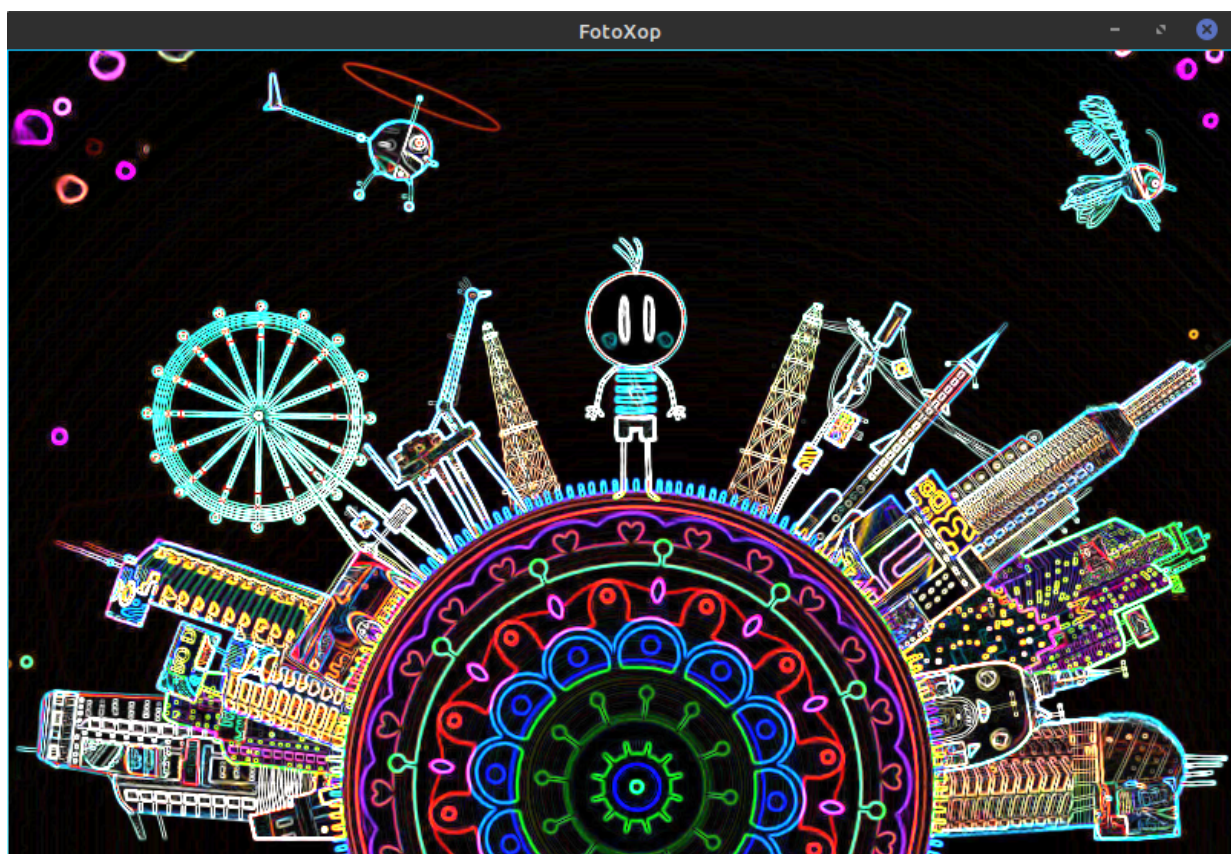


Figura 5: Filtro de Sobel (o resultado depende de como foi feita a combinação dos valores)

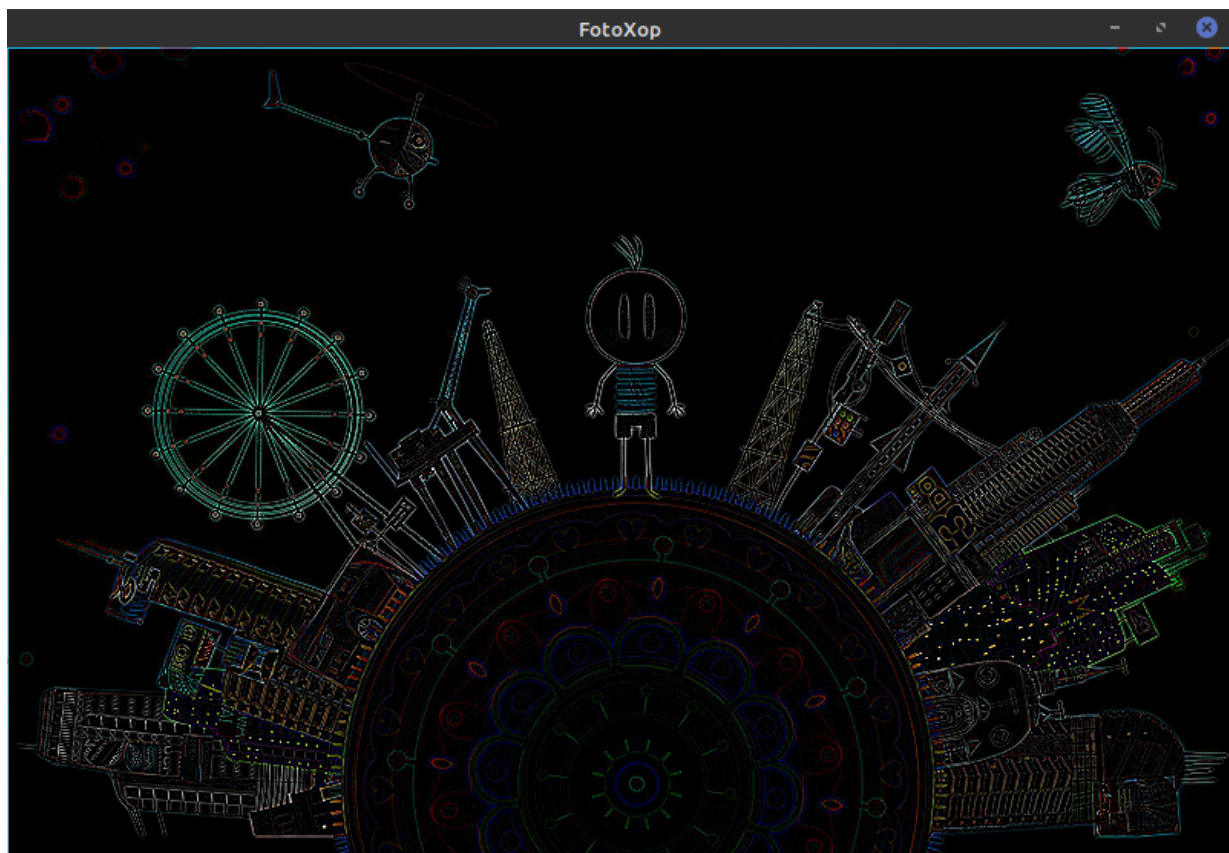


Figura 6: Detecção de bordas de Laplace