Convertendo imagens coloridas para P&B

Você recebeu um código Python que converte uma imagem colorida em tons de cinza (preto e branco) manualmente, pixel por pixel. O programa solicita ao usuário que selecione uma imagem, aplica a conversão usando a fórmula de luminância (0.299 * R + 0.587 * G + 0.114 * B) e salva o resultado. No entanto, para imagens grandes, o processo pode ser lento, pois cada pixel é processado sequencialmente.

Para melhorar o desempenho do programa, você deve paralelizar a conversão dos pixels utilizando threads. A ideia é dividir a imagem em partes menores e processar cada parte em uma thread separada, aproveitando melhor os recursos computacionais disponíveis.

Objetivo

Modifique o código fornecido para implementar a conversão de imagens em threads separadas . Cada thread deve processar uma faixa horizontal ou vertical da imagem, aplicando a mesma fórmula de luminância. Ao final, as partes processadas devem ser combinadas para formar a imagem completa em preto e branco.

Requisitos

1. Use a biblioteca threading do Python para criar threads.
2. Divida a imagem em faixas horizontais ou verticais e atribua cada faixa a uma thread diferente.
3. Garanta que todas as threads terminem antes de salvar a imagem final.
4. Certifique-se de que as threads não conflitem ao acessar ou modificar partes da imagem.
5. Combine as faixas processadas corretamente para gerar a imagem final.
6. Mantenha a estrutura modular do código original, criando funções específicas para cada tarefa.
7. Adicione uma função principal (main) que inicie as threads e aguarde sua conclusão.
8. A imagem convertida deve ser salva no formato escolhido pelo usuário (JPEG, PNG, etc.).
9. O programa deve exibir uma mensagem indicando o caminho onde a imagem foi salva.
10. Execute o programa com diferentes imagens e verifique se a conversão é feita corretamente.
11. Meça o tempo de execução com e sem threads para comparar o desempenho.

Código a ser paralelizado

```
from PIL import Image
from tkinter import Tk, filedialog

def converter_para_preto_e_branco_manual():

    try:
        root = Tk()
        root.withdraw()

        caminho_imagem = filedialog.askopenfilename(
            title="Selecione uma imagem",
            filetypes=[("Imagens", "*.jpg *.jpeg *.png *.bmp *.gif"), ("Todos os arquivos", "*.*")]
        )

        if not caminho_imagem:
            print("Nenhuma imagem foi selecionada.")
            return

        imagem = Image.open(caminho_imagem)
```

```
        imagem = imagem.convert("RGB")  # Garante que a imagem esteja no modo RGB
        largura, altura = imagem.size
        imagem_preto_branco = Image.new("L", (largura, altura))

        # Itera sobre cada pixel da imagem

        for x in range(largura):
            for y in range(altura):
                r, g, b = imagem.getpixel((x, y))
                luminancia = int(0.299 * r + 0.587 * g + 0.114 * b)
                imagem_preto_branco.putpixel((x, y), luminancia)

        caminho_saida = filedialog.asksaveasfilename(
            title="Salvar imagem em preto e branco",
            defaultextension=".jpg",
            filetypes=[("JPEG", "*.jpg"), ("PNG", "*.png"), ("Todos os arquivos", "*.*")]
        )

        if not caminho_saida:
            print("Operação de salvamento cancelada.")
            return

        # Salva a imagem em preto e branco no caminho especificado

        imagem_preto_branco.save(caminho_saida)
        print(f"Imagem convertida com sucesso! Salva em: {caminho_saida}")

    except Exception as e:
        print(f"Erro ao processar a imagem: {e}")

# Exemplo de uso

if __name__ == "__main__":
    converter_para_preto_e_branco_manual()
```