Dokumentace úlohy číslo 77 Konstrukce histogramu pro rastr v rámci zkoušky z předmětu PřfUK Úvod do programování (ZS 2023/24)

Konstrukce histogramu pro rastr

Pro vstupní rastr ve formátu JPG, PNG, GIF vytvořte histogram znázorňující absolutní četnost zastoupení jednotlivých barev, a to po jejich R, G, B složkách. Barevnou informaci v každém pixelu dekomponujte na RGB hodnoty v intervalu <0,255>. Pro vykreslení histogramu použijte Vámi zvolenou grafickou knihovnu.

Cílem tohoto kódu je vytvořit histogram základních barev rastrového obrázku. Histogram je grafická reprezentace, která ukazuje distribuci intenzity barev v obrázku. Každý kanál barvy (červená, zelená a modrá) je zpracován samostatně.

Existující algoritmy

Existují různé metody pro vytvoření histogramu barev. Nejběžnější metoda je first-fit bin packing, rozdělit každý kanál barvy na několik binů (např. 256 pro 8bitové obrázky) a spočítat počet pixelů v každém binu. Mezi další metody patří třeba Bucket sort.

Zvolený algoritmus

Tento kód používá knihovnu Pillow na převod rastru do bitmapy pro jednodušší práci s výpočtem. Tento kód využívá právě algoritmu first-fit bin packing. Knihovna Matplotlib je poté využita na samotné vykreslení grafu.

Struktura programu

Nejdříve na začátku kódu importujeme používané knihovny.

```
# Import necessary libraries
from PIL import Image
import matplotlib.pyplot as plt
```

Poté je kód strukturován v rámci zásad objektově orientovaného programování (OOP) do dvou tříd `Picture` a `Histogram`.

Třída `Picture` slouží k otevření obrázku a jeho převod na bitmapu.

```
# Define class Picture that reads an image and converts it to a BMP file
class Picture:
    # Define the __init__ method to read the image
    def __init__(self, image_path):
        self.image = Image.open(image_path)
    # Define a method to convert the image to a BMP file
    def convert_to_bmp(self):
        rgb_img = self.image.convert('RGB')
        rgb_img.save('./histogram/image.bmp')
```

Definujeme třídu `Histogram`, která přečte bitmapový obrázek a připraví prázdný list na samotné hodnoty pixelů.

```
class Histogram():
    def __init__(self):
        self.filename = './histogram/image.bmp'
        self.pixel_values = []
```

Definujeme metodu `read_bmp` na přečtení bitmapového obrázku.

```
# Define a method to read the BMP file
def read_bmp(self):
    with open(self.filename, 'rb') as f:
```

Pstupně čteme bajty bitmapového obrázku. Nejprve zahazujeme záhlaví a poté čteme záhlaví s informacemi o obrázku, ze kterých vyčteme šířku a výšku obrázku. Poté zahazujeme zbytek záhlaví.

```
f.read(14) # Skip bitmap file header
bmp_header = f.read(40) # Read bitmap info header
# Extract image width and height from the header
width, height = (
    int.from_bytes(bmp_header[4:8],
        'little'),
        int.from_bytes(bmp_header[8:12],
        'little')
        )
    # Skip the rest of the header
    f.read(18)
```

Čteme hodnoty pixelů řádek po řádku, pomocí informace o výšce a šířce obrázku. V bitmapovém obrázku jsou hodnoty RGB uloženy v pořadí GBR. Čteme tedy bajt po bajtu a tyto bajty převádíme na integery. Řádky poté ukládáme do 2D listu. Po každém řádku zahazujeme bajty, které oddělují hodnoty pixelů.

```
# Read the pixel values row by row
for _ in range(height):
    row = []
    for _ in range(width):
        # Read the pixel values
        b = int.from_bytes(f.read(1), 'little')
        g = int.from_bytes(f.read(1), 'little')
        r = int.from_bytes(f.read(1), 'little')
        row.append((r, g, b))
    self.pixel_values.append(row)
    # Skip any padding bytes
    f.read(width % 4)
```

Definujeme metodu `plot` na vykreslení grafu. Nejprve z 2D listu `pixel_values` vytřídíme jednotlivé barvy.

```
# Define a method to plot the histogram
def plot(self):
    # Separate the pixel data into three lists, one for each color channel
    red_values = [pixel[0] for row in self.pixel_values for pixel in row]
    green_values = [pixel[1] for row in self.pixel_values for pixel in row]
    blue_values = [pixel[2] for row in self.pixel_values for pixel in row]
```

Poté listy s informacemi o hodnotách barev vykreslíme pomocí `Matplolib`.

```
# Plot a histogram for each color channel
plt.hist(red_values, bins=256, color='red', alpha=0.5)
plt.hist(green_values, bins=256, color='green', alpha=0.5)
plt.hist(blue_values, bins=256, color='blue', alpha=0.5)

# Add a title and labels to the plot
plt.title('Color Histogram')
plt.xlabel('Color Value')
plt.ylabel('Pixel Count')

plt.show()
```

Datové struktury

Hlavní datovou strukturou je 2D list, který reprezentuje hodnoty pixelů. Každý pixel je reprezentován trojicí hodnot (R, G, B). Další datovou strukturou je list využitý na uložení těchto tří barev samostatně.

Vstupní/výstupní data

Vstupní data jsou obrázek ve formátu JPG, PNG, nebo GIF. Výstupem je histogram zobrazený pomocí Matplotlib.

Problematická místa kódu

Kód používá pevně zadané cesty k obrázkům, to zhoršuje jeho flexibilitu. Zároveň kód neefektivně využívá paměť při čtení obrázku. Je to poznat při spouštění kódu a může to teoreticky způsobit problém při využití na větší obrázky.

Možná vylepšení

Kód by mohl být opatřen dotazováním uživatele na cestu k obrázku a algoritmus by mohl být upraven pro efektivnější využití paměti.