Relatório Técnico do Projeto: Extração de Características e Criação de Mosaico

Desenvolvido em Python e C com OpenCV

1 Introdução

Este relatório apresenta um projeto composto por duas partes interligadas:

1. **Extração de Características:** Um código em Python que utiliza OpenCV e NumPy para extrair características de imagens, aplicando técnicas como Cadeia de Código (Code Chain) e Padrões Binários Locais (LBP).
2. **Criação do Mosaico:** Um código em C que, também utilizando OpenCV, gera um mosaico a partir de uma imagem original, substituindo suas regiões por pequenos tiles cujas médias de cor foram previamente calculadas.

O objetivo é demonstrar, de forma integrada, o uso de técnicas de processamento de imagens para análise de padrões e para a criação de composições visuais artísticas.

2 Objetivo do Projeto

O projeto visa:

- Extrair características relevantes de imagens, representando informações sobre textura e estrutura.
- Gerar um banco de dados (arquivo **mosaic.txt**) contendo as médias de cor dos tiles.
- Criar um mosaico, substituindo regiões de uma imagem original pelos tiles que melhor se aproximam, em termos de cor, da média de cada região.

3 Arquivos e Componentes

O projeto é composto pelos seguintes arquivos:

- **extrai_caract.py**: Código em Python para extração de características das imagens.
- **fazmosaico.c**: Código em C que cria o mosaico, dividindo a imagem original e substituindo as regiões pelos tiles adequados.
- **mosaic.c**: Código em C que processa os tiles (imagens individuais) e gera o arquivo de dados **mosaic.txt** com as médias de cor.

- **mosaic.txt**: Arquivo de texto contendo linhas no formato `nome_imagem.jpg:valor_red:valor_gre`
- **nova.jpg**: Imagem resultante do processo de criação do mosaico.

4 Funcionamento do Código

4.1 Extração de Características (Python)

Esta parte do projeto utiliza OpenCV e NumPy para extrair características de imagens. As principais funções são:

4.1.1 HistCodeChain

Esta função aplica um limiar para converter a imagem em binária, inverte os valores, detecta os contornos e calcula a "cadeia de código" para representar a direção dos movimentos entre pixels consecutivos nos contornos. Ao final, gera um histograma normalizado com 8 posições, representando as frequências de cada direção.

```
def HistCodeChain(img):
    ret, thresh = cv2.threshold(img, 127, 255, 0)
    thresh = abs(thresh - 255)
    contours, hierarchy = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
        cv2.CHAIN_APPROX_NONE)
    codeChains = []
    for o, objeto in enumerate(contours):
        for i, element in enumerate(objeto):
            codeChains.append([])
            if i != 0:
                disX = objeto[i][0][1] - objeto[i-1][0][1]
                disY = objeto[i][0][0] - objeto[i-1][0][0]
                if disX == 1 and disY == 0:
                    codeChains[o] = np.append(codeChains[o], 0)
                if disX == 1 and disY == 1:
                    codeChains[o] = np.append(codeChains[o], 1)
                if disX == 0 and disY == 1:
                    codeChains[o] = np.append(codeChains[o], 2)
                if disX == -1 and disY == 1:
                    codeChains[o] = np.append(codeChains[o], 3)
                if disX == -1 and disY == 0:
                    codeChains[o] = np.append(codeChains[o], 4)
                if disX == -1 and disY == -1:
                    codeChains[o] = np.append(codeChains[o], 5)
                if disX == 0 and disY == -1:
                    codeChains[o] = np.append(codeChains[o], 6)
                if disX == 1 and disY == -1:
                    codeChains[o] = np.append(codeChains[o], 7)
    if len(codeChains) <= 0:
        return np.zeros(8)
    histogramas = np.zeros((len(codeChains), 8))
    for i, cc in enumerate(codeChains):
        for j in range(8):
            histogramas[i][j] = np.count_nonzero(cc == j)
    histograma = np.zeros(8)
```

```
    for h in histogramas:
        histograma += h
    soma = histograma.sum()
    if soma != 0:
        histograma /= soma
    return histograma
```

4.1.2 Local Binary Pattern (LBP)

A função `calcula_pixel_lbp` calcula o código LBP para cada pixel, comparando o valor do pixel central com os de seus 8 vizinhos. Cada comparação gera um bit que, ponderado por uma potência de 2, forma um valor entre 0 e 255 para representar a textura local.

```
def calcula_pixel_lbp(img, x, y):
    center = img[x][y]
    val_ar = []
    val_ar.append(pega_pixel(img, center, x-1, y+1)) # acima direita
    val_ar.append(pega_pixel(img, center, x, y+1)) # direita
    val_ar.append(pega_pixel(img, center, x+1, y+1)) # abaixo direita
    val_ar.append(pega_pixel(img, center, x+1, y)) # abaixo
    val_ar.append(pega_pixel(img, center, x+1, y-1)) # abaixo esquerda
    val_ar.append(pega_pixel(img, center, x, y-1)) # esquerda
    val_ar.append(pega_pixel(img, center, x-1, y-1)) # acima esquerda
    val_ar.append(pega_pixel(img, center, x-1, y)) # acima
    power_val = [1, 2, 4, 8, 16, 32, 64, 128]
    val = 0
    for i in range(len(val_ar)):
        val += val_ar[i] * power_val[i]
    return val
```

O código percorre as imagens do conjunto de treinamento, processa cada imagem para extrair o histograma do LBP e o histograma da Cadeia de Código, e armazena os vetores de características para posterior análise.

4.2 Criação do Mosaico (C)

A segunda parte do projeto utiliza C e OpenCV para gerar a imagem em mosaico. Essa parte é dividida em dois programas:

1. **mosaic.c**: Processa uma sequência de imagens (tiles) e calcula a média dos valores de cor (RGB) de cada tile. Os resultados são armazenados no arquivo **mosaic.txt**.

2. **fazmosaico.c**: Lê a imagem original e o arquivo **mosaic.txt** para, região por região, substituir partes da imagem original pelo tile cujo perfil de cor se aproxima mais da média daquela região.

4.2.1 Funcionamento da Preparação do Banco de Dados

O programa **mosaic.c** realiza as seguintes etapas:

- Abre imagens nomeadas sequencialmente (por exemplo, `1.jpg`, `2.jpg`, etc.).

- Calcula a média dos valores de cor (vermelho, verde e azul) de cada imagem utilizando a função `encontrar_car`.
- Grava uma linha no arquivo **mosaic.txt** no formato:

  `nome_imagem.jpg:red:green:blue`

4.2.2 Funcionamento da Criação do Mosaico

O programa **fazmosaico.c** é executado com parâmetros via linha de comando. Por exemplo:

```
./fazmosaico 14.jpg ci067 mosaic.txt 1 10
```

Onde:

- `14.jpg` é a imagem de entrada.
- `ci067` é um parâmetro auxiliar, representando a pasta onde estão os tiles.
- `mosaic.txt` é o arquivo contendo os dados dos tiles.
- `1` e `10` são parâmetros numéricos; o último (10) define a quantidade de células em cada dimensão da matriz (por exemplo, uma divisão 10x10).

O processo de criação do mosaico ocorre conforme descrito:

1. A imagem original é dividida em uma matriz de regiões.
2. Para cada região, é calculada a média dos valores de cor.
3. Utilizando o arquivo **mosaic.txt**, o programa seleciona o tile cuja média de cor é mais próxima da média da região.
4. A função `subimg` redimensiona o tile selecionado e o insere na posição correspondente da imagem original.

Ao final, a imagem resultante é salva como **nova.jpg**.

5 Compilação e Execução

5.1 Extração de Características (Python)

Certifique-se de ter o OpenCV e o NumPy instalados. Em sistemas Linux, por exemplo:

```
sudo apt-get install python-opencv python-numpy
```

Para executar o código de extração de características:

```
python extrai_caract.py -t caminho/para/treino -l arquivo_de_labels.txt > saida_treino
```

5.2 Geração do Banco de Dados (C)

Compile o programa **mosaic.c**:

```
gcc `pkg-config --cflags --libs opencv` -o mosaic mosaic.c
```

Execute-o para gerar o arquivo **mosaic.txt**:

```
./mosaic mosaic.txt
```

5.3 Criação do Mosaico (C)

Compile o programa **fazmosaico.c**:

```
gcc `pkg-config --cflags --libs opencv` -o fazmosaico fazmosaico.c
```

Execute-o, por exemplo:

```
./fazmosaico 14.jpg ci067 mosaic.txt 1 10
```

A imagem resultante será salva como **nova.jpg**.

6 Conclusão

O projeto integra técnicas de processamento de imagens para extrair características relevantes (usando Python, OpenCV e NumPy) e para gerar uma composição visual (mosaico) a partir dessas informações (usando C e OpenCV).

- A extração de características possibilita a análise de texturas e padrões estruturais.
- A criação do mosaico demonstra a aplicação prática desses dados, substituindo regiões da imagem original por tiles que preservam a tonalidade e a estrutura geral.

Esse conjunto de técnicas pode ser aplicado em reconhecimento de padrões, classificação de imagens e em projetos artísticos de transformação de imagens.