Processamento Digital de Imagens

Table of Contents

1. Manipulando pixels em uma imagem.

A manipulação de pixels em uma imagem refere-se ao processo de alterar as propriedades dos pixels individuais que compõe a imagem, como, por exemplo, modificar o valor da cor do pixel, alterar a sua posição na imagem, aplicar filtros ou efeitos especiais, entre outros. Desta forma, é possível realizar uma infinidade de tarefas, como redimensionar uma imagem, remover objetos indesejados, corrigir imperfeições, aplicar efeitos artísticos, criar animações, entre outras aplicações criativas e práticas.

1.1. Filtro Negativo (regions.cpp)

Um filtro negativo em uma imagem é uma técnica de manipulação de pixels que inverte as cores da imagem original. Nesse filtro, cada pixel da imagem é transformado em seu complemento,

resultando em uma imagem com cores invertidas. Além de criar um efeito estético interessante, o filtro negativo também pode ser útil em certas aplicações, como melhorar a visualização de detalhes em imagens com alto contraste ou realçar certos elementos. No entanto, é importante notar que a aplicação de um filtro negativo em uma imagem não é uma técnica que preserva informações importantes da imagem original, mas sim uma transformação visual que pode ser usada para efeitos artísticos ou estilísticos.

1.1.1. Código & Resultado.

regions.cpp

```
#include <iostream>
#include <opencv2/opencv.hpp>

using namespace cv;
using namespace std;

int main() {
    cout << "INICIANDO..." << endl;

    Mat image;
    int P1X, P1Y, P2X, P2Y;
    char diretorio[1000];

    cout << "Digite a loc da imagem: " << endl;
    cout << "EX.: C:\\User\\Desktop\\<nomedaimagem>.png" << endl;
    cin >> diretorio;

    auto imageO = imread(diretorio);
    image = imread(diretorio, IMREAD_GRAYSCALE);

    int rows = image.rows;
    int cols = image.cols;

    cout << "Digite o ponto P1 da imagem" << endl;
    cout << "Entre Linha 0 e" << " " << (rows - 1) << " " << "e Coluna 0 e" << " " <<
(cols - 1) << endl;
    cin >> P1X >> P1Y;
    cout << "(" << P1X << "," << P1Y << ")" << endl;
    cout << "Digite 0 para cancelar e 1 para confirmar" << endl;

    int confirmaP1;

    do {
        cin >> confirmaP1;
        if (confirmaP1 == 0) {
            cout << "Digite o ponto P1 da imagem" << endl;
            cout << "Entre Linha 0 e" << " " << (rows - 1) << " " << "e Coluna 0 e" <<
" " << (cols - 1) << endl;
            cin >> P1X >> P1Y;
```

```
            cout << "(" << P1X << "," << P1Y << ")" << endl;

            cout << "Digite 0 para cancelar e 1 para confirmar" << endl;
            cin >> confirmaP1;
        }

        if (confirmaP1 == 1 && (P1X < 0 || P1X > rows)) {
            cout << "Valor de X do ponto P1 localiza-se fora da imagem, digite
novamente P1X: " << endl;
            cin >> P1X;

            cout << "(" << P1X << "," << P1Y << ")" << endl;
            cout << "Digite 0 para cancelar e 1 para confirmar" << endl;
            cin >> confirmaP1;
        }

        if (confirmaP1 == 1 && (P1Y< 0 || P1Y > cols)) {
            cout << "Valor de Y do ponto P1 localiza-se fora da imagem, digite
novamente P1Y: " << endl;
            cin >> P1Y;

            cout << "(" << P1X << "," << P1Y << ")" << endl;
            cout << "Digite 0 para cancelar e 1 para confirmar" << endl;
            cin >> confirmaP1;
        }

    } while (confirmaP1 == 0);

    cout << "Digite o ponto P2 da imagem" << endl;
    cout << "Entre Linha 0 e" << " " << (rows - 1) << " " << "e Coluna 0 e" << " " <<
(cols - 1) << endl;
    cin >> P2X >> P2Y;
    cout << "(" << P2X << "," << P2Y << ")" << endl;
    cout << "Digite 0 para cancelar e 1 para confirmar" << endl;

    int confirmaP2;
    do {
        cin >> confirmaP2;
        if (confirmaP2 == 0) {
            cout << "Digite o ponto P2 da imagem" << endl;
            cout << "Entre Linha 0 e" << " " << (rows - 1) << " " << "e Coluna 0 e" <<
" " << (cols - 1) << endl;
            cin >> P2X >> P2Y;
            cout << "(" << P2X << "," << P2Y << ")" << endl;

            cout << "Digite 0 para cancelar e 1 para confirmar" << endl;
            cin >> confirmaP2;
        }

        if (confirmaP2 == 1 && (P2X < 0 || P2X > rows || P2X < P1X)) {
            cout << "Valor de X do ponto P2 localiza-se fora da imagem ou eh menor que
```

```
P1X, digite novamente P2X: " << endl;
            cin >> P2X;

            cout << "(" << P2X << "," << P2Y << ")" << endl;
            cout << "Digite 0 para cancelar e 1 para confirmar" << endl;
            cin >> confirmaP2;
        }

        if (confirmaP2 == 1 && (P2Y< 0 || P2Y > cols || P2Y < P1Y)) {
            cout << "Valor de Y do ponto P2 localiza-se fora da imagem ou eh menor que
P1Y, digite novamente P2Y: " << endl;
            cin >> P2Y;

            cout << "(" << P2X << "," << P2Y << ")" << endl;
            cout << "Digite 0 para cancelar e 1 para confirmar" << endl;
            cin >> confirmaP2;
        }
    } while (confirmaP2 == 0);

    if (!image.data) {
        cout << "Imagem nao encontrada!" << endl;
    }

    for (int i = P1X; i < P2X; i++) {
        for (int j = P1Y; j < P2Y; j++) {

            image.at<uchar>(i, j) = 255 - image.at<uchar>(i, j);

        }
    }

    imwrite("janelaNegativo.png", image);
    namedWindow("janelaOriginal", WINDOW_AUTOSIZE);
    imshow("janelaOriginal", image0);
    namedWindow("janelaNegativo", WINDOW_AUTOSIZE);
    imshow("janelaNegativo", image);

    waitKey();

    return 0;

}
```

1.2 Troca de regiões

1.2.1 Código & Resultado.

2. Serialização via FileStore

2.1. filestore.cpp

2.1.1 Código & Resultado.

3. Decomposição de imagens em planos de bits

3.1 Esteganografia em imagens digitais

3.1.1 Código & Resultado.

4. Preenchendo regiões

4.1 FloodFill

4.1.1 Código & Resultado.

5. Manipulação de histogramas

5.1 Histograma

5.1.1 Código & Resultado

6. Filtragem no domínio espacial I

6.1 Filtro Espacial

6.1.1 Código & Resultado

7. Filtragem no domínio espacial II

7.1 TiltShift

7.1.1 Código & Resultado

8. A Tranformada Discreta de Fourier

8.1

8.1.1 Código & Resultado

9. Filtragem no Domínio da Frequência

9.1 Filtro Homomórfico

9.1.1 Código & Resultado

10. Detecção de bordas com o algoritmo de Canny

10.1 Canny & Pontilhismo

10.1.1 Código & Resultado

11. Quantização vetorial com k-means

11.1 K-means

11.1.1 Código & Resultado