Processamento Digital de Imagens

Table of Contents

1.	Manipulando pixels em uma imagem	. 1
	1.1 Exercicio 1.1 - Filtro Negativo (regions.cpp)	. 1
	1.1.2 Exercicio 1.2 - Troca Regiões (trocaregioes.cpp)	. 3
2.	Serialização de dados em ponto flutuante via FileStorage	. 3
	2.1. Exercicio 2 - filestorage.cpp	. 3
3.	Decomposição de imagens em planos de bits	. 3
	3.1. Exercicio 3 - esteg-encode.cpp	. 3
4.	Preenchendo regiões	. 3
	4.1. Exercicio 4 - labeling.cpp	. 3
5.	Manipulação de histogramas	. 4
	5.1. Exercicio 5 - histogram.cpp	. 4
6.	Filtragem no domínio espacial I	. 4
	6.1. Exercicio 6 - filtroespacial.cpp	. 4
7.	Filtragem no domínio espacial II	. 4
	7.1. Exercicio 7 - addweighted.cpp	. 4
8.	A Tranformada Discreta de Fourier	. 4
	8.1. Exercicio 8 - dftimage.cpp	. 4

1. Manipulando pixels em uma imagem

Manipular pixels em uma imagem envolve a capacidade de alterar individualmente os elementos de cor que compõem a imagem. Cada pixel contém informações sobre sua cor específica, como vermelho, verde e azul (RGB), além de valores de transparência em alguns casos. Ao manipular os pixels, é possível realizar uma variedade de transformações na imagem, como ajustar o brilho, a saturação, o contraste, aplicar filtros, redimensionar ou recortar. Essas manipulações permitem corrigir imperfeições, realçar detalhes, criar efeitos especiais, entre outras possibilidades. A manipulação de pixels é uma técnica fundamental em áreas como processamento de imagem, design gráfico, edição de fotos e criação de arte digital. Com a ajuda de bibliotecas de processamento de imagem, é possível acessar e modificar os valores dos pixels em uma imagem, abrindo caminho para inúmeras possibilidades criativas e práticas.

1.1 Exercicio 1.1 - Filtro Negativo (regions.cpp)

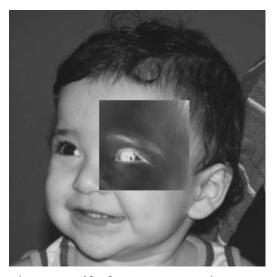


Figure 1. saída do programa regions.cpp

1.1.1. Código e Resultado.

Incluindo codigo do exemplo regions.

regions.cpp

```
#include <iostream>
#include <opencv2/opencv.hpp>
using namespace cv;
using namespace std;
int main(int, char** argv) {
    Mat image;
    image=imread(argv[1], IMREAD_GRAYSCALE);
    if (image.empty()) {
        cout << "Imagem não foi carregada" << endl;</pre>
        return 1;
    }
    Point p1, p2;
    cout << "Tamanho da imagem: " << image.rows << "x" << image.cols << endl;</pre>
    cout << "Digite as coordenadas do ponto P1:" << endl;</pre>
    cin >> p1.y >> p1.x;
    do{
    cout << "Digite as coordenadas do ponto P2:" << endl;</pre>
    cin >> p2.y >> p2.x;
    if (p1.y >= p2.y || p1.x >= p2.x){
        cout << "Ambas coordenadas do ponto P2 tem que ser maiores que as coordenadas</pre>
do ponto P1, escreva novamente!"<<endl;</pre>
```

```
}while(p1.y >= p2.y || p1.x >= p2.x);

for (int i = p1.x; i < p2.x; i++) {
      for (int j = p1.y; j < p2.y; j++) {
         image.at<uchar>(i, j) = 255 - image.at<uchar>(i, j);
      }
}

namedWindow("janela", WINDOW_AUTOSIZE);
imshow("janela", image);
waitKey(0);
imwrite("Regions.png", image);
return 0;
}
```

- 1.1.2 Exercicio 1.2 Troca Regiões (trocaregioes.cpp)
- 2. Serialização de dados em ponto flutuante via FileStorage
- 2.1. Exercicio 2 filestorage.cpp
- 2.1.1. Código e Resultado.
- 3. Decomposição de imagens em planos de bits
- 3.1. Exercicio 3 esteg-encode.cpp
- 3.1.1. Código e Resultado.
- 4. Preenchendo regiões
- 4.1. Exercicio 4 labeling.cpp
- 4.1.1. Código e Resultado.

5. Manipulação de histogramas

5.1. Exercicio 5 - histogram.cpp

5.1.1. Código e Resultado.

6. Filtragem no domínio espacial I

6.1. Exercicio 6 - filtroespacial.cpp

6.1.1. Código e Resultado.

7. Filtragem no domínio espacial II

7.1. Exercicio 7 - addweighted.cpp

7.1.1. Código e Resultado.

8. A Tranformada Discreta de Fourier

8.1. Exercicio 8 - dftimage.cpp

8.1.1. Código e Resultado.