Работа K. фильтрация изображений

автор: Дворядкин К.А. дата: 2022-05-16T15:20:32 хранилище:
https://github.com/DiorChoppa/imageprocessing-spring-2022/blob/main/prj.labs/K/K.cpp

Задание

0. текст, иллюстрации и подписи отчета придумываем самостоятельно
1. нарисовать

- одноканальное изображение
- поле 2x3 из квадратных клеток 150x150px черного, белого и серого (127) цвета (соседние цвета разные)
- в центре клеток - круг другого цвета (все сочетания перебрать)

2. отфильтровать и визуализировать I1 (фильтр вида) 1 0 0 -1
3. отфильтровать и визуализировать I2 (фильтр вида) 0 1 -1 0
4. вычислить и визуалиировать геометрическое среднее (корень из суммы квадратов I1 и I2)

Первоначально нам нужно задать исходное изображение (круги рисуем с помощью метода circle):

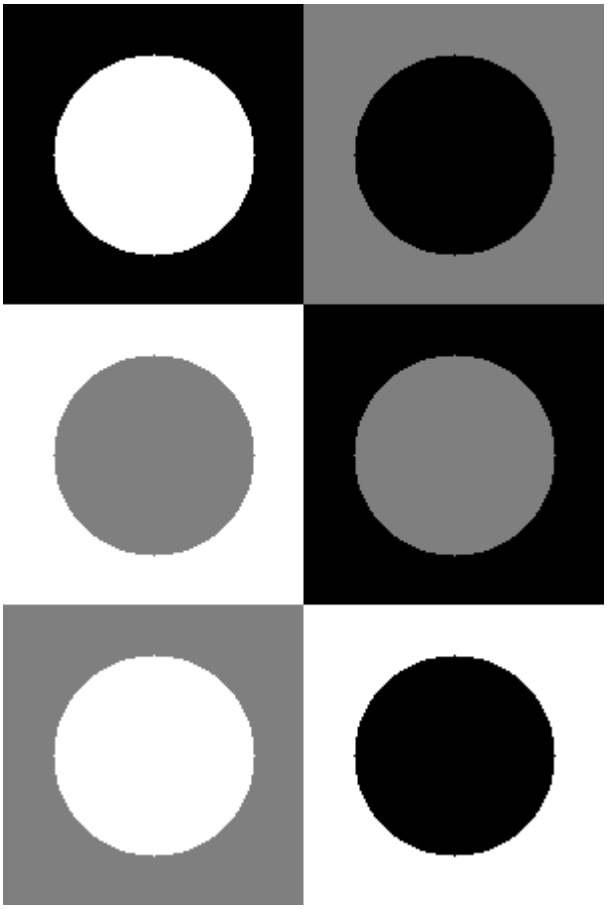

Рис. 1. Исходное изображение 2x3.

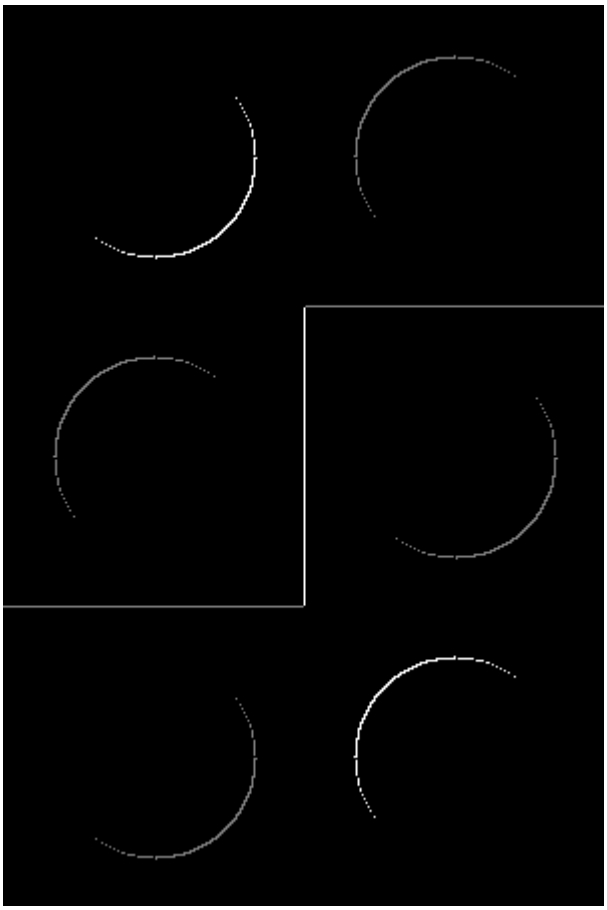

Рис. 2. Изображение, полученное в результате применения метода filter2d с параметрами kernel = [1,0,0,-1] и anchor cv::Point(-1, -1)

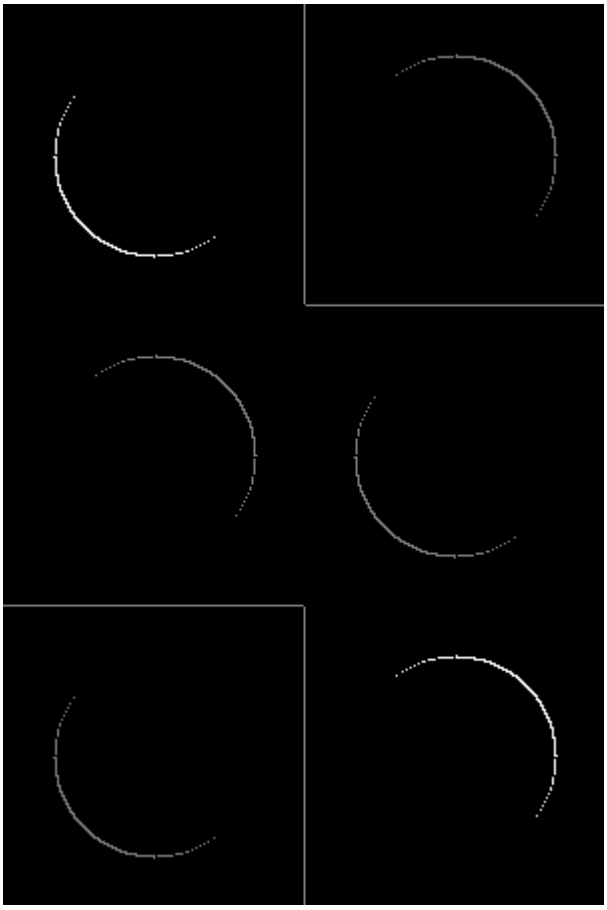

Рис. 3. Изображение, полученное в результате применения метода filter2d с параметрами kernel = [0,1,-1,0] и anchor cv::Point(-1, -1)

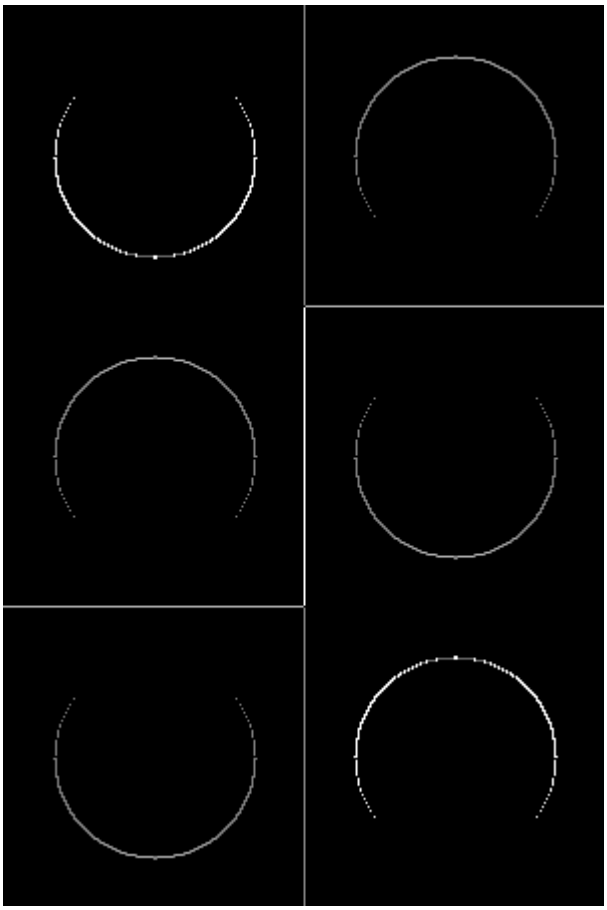

Рис. 4. Изображение, полученное в результате применения геометрического среднего из Рис. 2 и Рис. 3.

Результаты

Текст программы

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

int main() {

    // Drawing source image

    cv::Mat img(450, 300, CV_8UC1);
    for (int i = 0; i < 150; i++)
    {
        for (int j = 0; j < 150; j++)
        {
            img.at<uint8_t>(j, i) = 0;
            img.at<uint8_t>(j, i+150) = 127;
            img.at<uint8_t>(j+150, i) = 255;
            img.at<uint8_t>(j+300, i) = 127;
            img.at<uint8_t>(j+150, i+150) = 0;
            img.at<uint8_t>(j+300, i+150) = 255;
        }

    }
```

```
    cv::circle(img, cv::Point(75, 75), 50, 255, cv::FILLED);
    cv::circle(img, cv::Point(75+150, 75), 50, 0, cv::FILLED);
    cv::circle(img, cv::Point(75, 75+150), 50, 127, cv::FILLED);
    cv::circle(img, cv::Point(75+150, 75+150), 50, 127, cv::FILLED);
    cv::circle(img, cv::Point(75, 75+300), 50, 255, cv::FILLED);
    cv::circle(img, cv::Point(75+150, 75+300), 50, 0, cv::FILLED);

    cv::imwrite("K.png", img);
    cv::imshow("K", img);

    //I1

    cv::Mat kernel_f = (cv::Mat_<int>(2, 2) << 1, 0, 0, -1);
    cv::Mat I1;
    cv::filter2D(img, I1, -1, kernel_f, {-1, -1}, 0.0);
    cv::imwrite("I1.png", I1);
    cv::imshow("I1", I1);

    //I2

    cv::Mat kernel_s = (cv::Mat_<int>(2, 2) << 0, 1, -1, 0);
    cv::Mat I2;
    cv::filter2D(img, I2, -1, kernel_s, {-1, -1}, 0.0);
    cv::imwrite("I2.png", I2);
    cv::imshow("I2", I2);

    //Mean geometry

    cv::Mat gm(img);
    for(int i = 0; i < img.rows; i++){
        for(int j = 0; j < img.cols; j++){
            gm.at<uchar>(i, j) = sqrt(pow(I1.at<uchar>(i, j), 2) +
pow(I2.at<uchar>(i, j), 2));
        }
    }
    cv::imwrite("gm.png", gm);
    cv::imshow("gm", gm);

    cv::waitKey(0);

    return 0;
}
```