Работа 4. Использование частных производных для выделения границ

автор: Мартынов Д.А. url: https://gitlab.com/M_D_/image-processing

Задание

1. Сгенерировать серое тестовое изображение из квадратов и кругов с разными уровнями яркости (0, 127 и 255) так, чтобы присутствовали все сочетания.
2. Применить первый линейный фильтр и сделать визуализацию результата F_1.
3. Применить второй линейный фильтр и сделать визуализацию результата F_2.
4. Вычислить $R=\sqrt{F_1^2 + F_2^2}$ и сделать визуализацию R.

Результаты

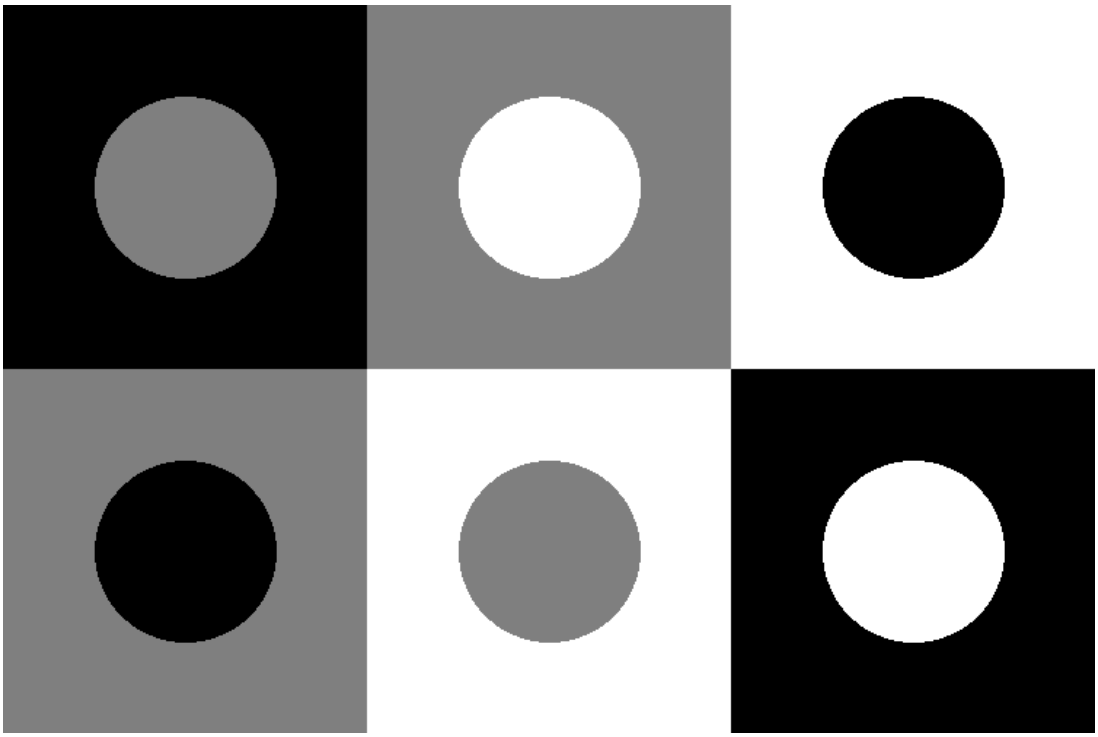

Рис. 1. Исходное тестовое изображение

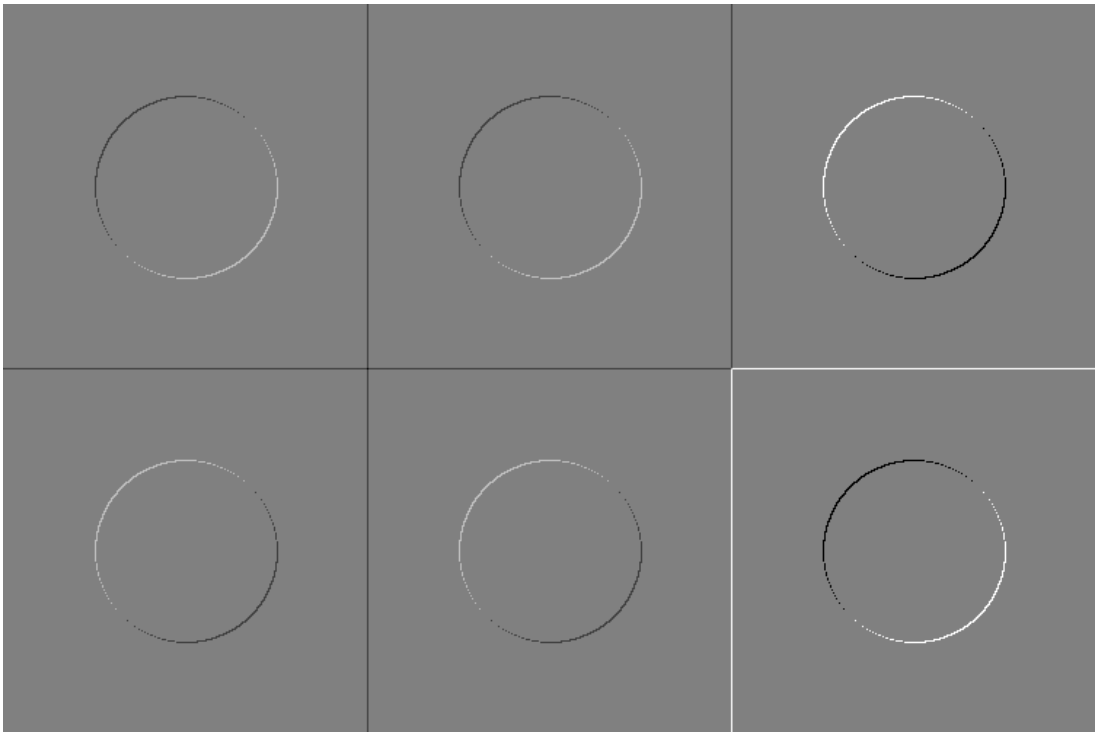

Рис. 2. Визуализация результата F_1 применения фильтра

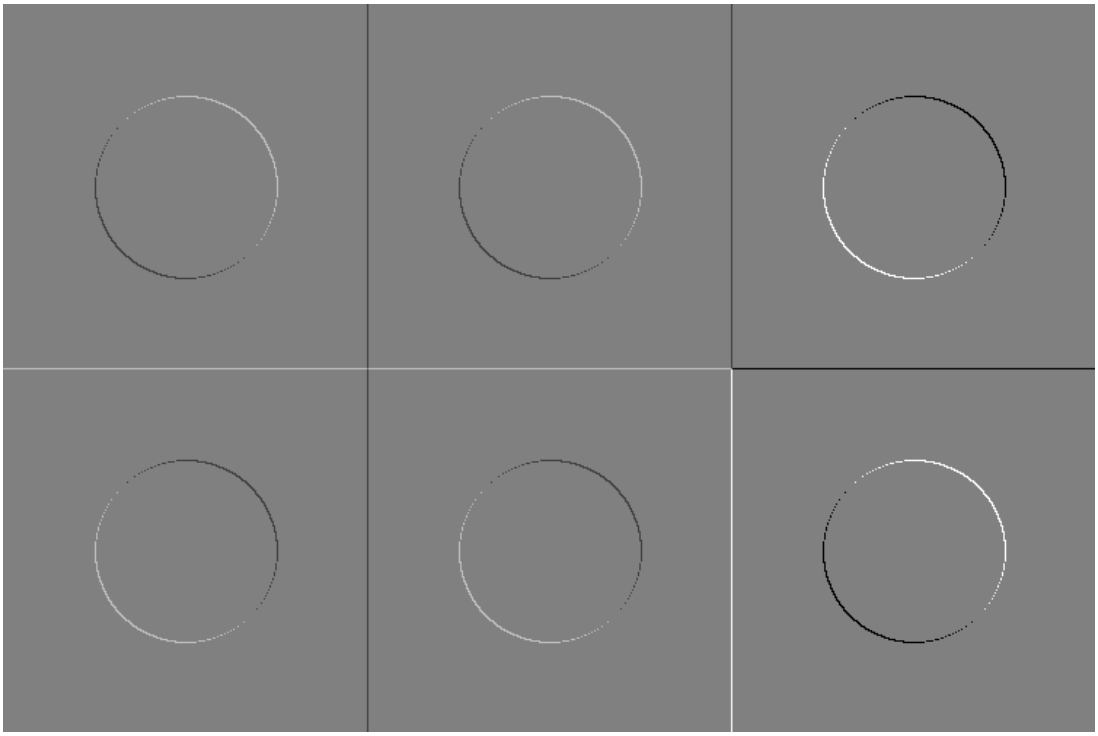

Рис. 3. Визуализация результата F_2 применения фильтра

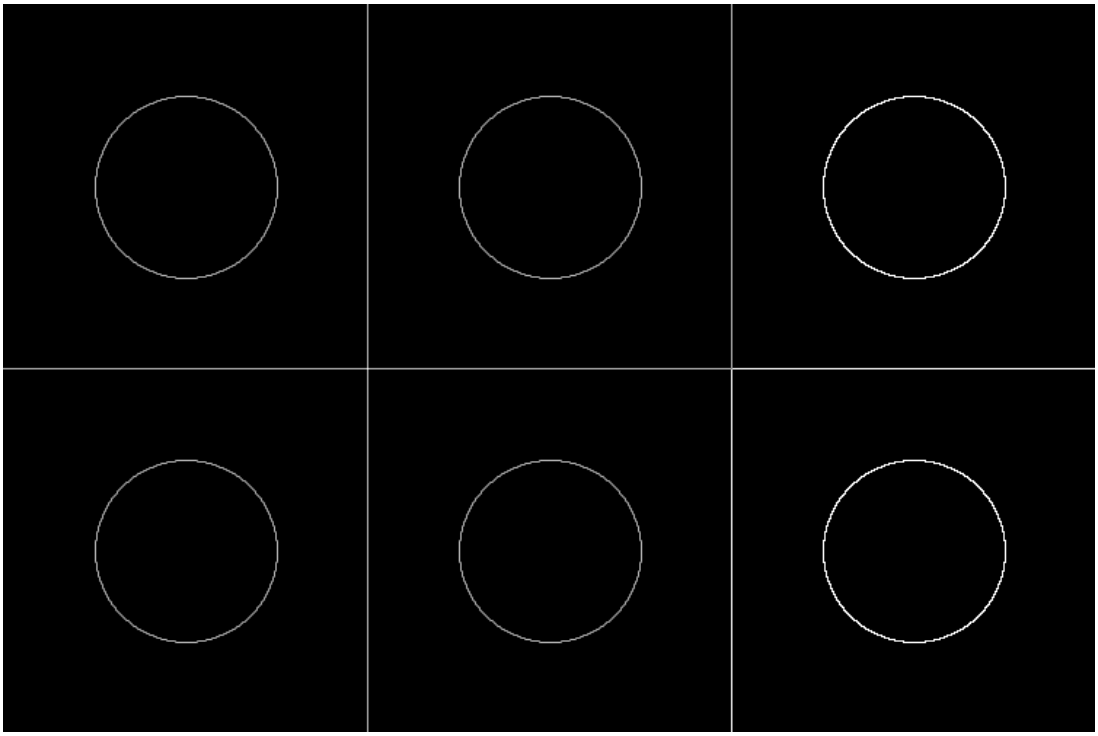

Рис. 4. Визуализация модуля градиента *R*

Текст программы

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

// Длина стороны квадрата в исходном изображении
constexpr int size{ 500 };

// Функция создания прямоугольника 3x2 из квадратов и кругов
cv::Mat rectCirc(cv::Mat &rectangle)
{

    std::vector<cv::Rect>split;
    for (int i{ 0 }; i < 2; ++i)
    {
        for (int j{ 0 }; j < 3; ++j)
            split.push_back(cv::Rect(j * size, i * size, size, size));
    }

    std::vector<cv::Point> center;
    for (int i{ 0 }; i < split.size(); ++i)
    {
        center.push_back((split[i].br() + split[i].tl()) * 0.5);
    }

    std::vector<int>color{ 0,127,127,255,255,0,127,0,255,127,0,255 };

    int k{ 0 };
    for (int i{ 0 }; i < 6; ++i)
    {
        cv::rectangle(rectangle, split[i], color[k], cv::FILLED, CV_8UC1, 0);
        cv::circle(rectangle, center[i], size / 4, color[k + 1], cv::FILLED, CV_8UC1, 0);
        k += 2;
    }
```

```
    return rectangle;
}

int main()
{
    // Создание картинки из квадратов в кругов
    cv::Mat rectangle(size * 2, size * 3, CV_8UC1);
    rectangle = rectCirc(rectangle);

    // Параметры фильтра
    cv::Point anchor = cv::Point(-1, -1); // якорь по центру ядра
    double delta = 127; // прибавить каждому пикселю 127
    int ddepth = -1;

    // Создание ядра свертки
    cv::Mat filter1(2, 2, CV_32F);
    filter1.at<float>(0, 0) = 1;
    filter1.at<float>(1, 0) = 0;
    filter1.at<float>(0, 1) = 0;
    filter1.at<float>(1, 1) = -1;

    // Создание второго ядра свертки
    cv::Mat filter2(2, 2, CV_32F);
    filter2.at<float>(0, 0) = 0;
    filter2.at<float>(1, 0) = 1;
    filter2.at<float>(0, 1) = -1;
    filter2.at<float>(1, 1) = 0;

    // Создание ядра свертки типа sqrt(a1^2+a2^2)
    /*cv::Mat filterSqrt(2, 2, CV_32F);
    cv::pow(filter1.mul(filter1) + filter2.mul(filter2), 0.5, filterSqrt);*/

    // Применение фильтров
    /*cv::Mat filtered1(rectangle.cols, rectangle.rows, CV_8UC1);
    cv::filter2D(rectangle, filtered1, ddepth, filter1, anchor, delta, cv::BORDER_DEFAULT);

    cv::Mat filtered2(rectangle.cols, rectangle.rows, CV_8UC1);
    cv::filter2D(rectangle, filtered2, ddepth, filter2, anchor, delta, cv::BORDER_DEFAULT);

    cv::Mat filteredBoth(rectangle.cols, rectangle.rows, CV_8UC1);
    cv::filter2D(rectangle, filteredBoth, ddepth, filterSqrt, anchor, delta, cv::BORDER_DEFAULT);*/

    // Вариант 2 ---------------------------------------------------------------------------

    // Два фидьтра применяются так же, как в первом способе
    // при этом нормализация производится после применения фильтра, а не во время
    cv::Mat filtered1(rectangle.cols, rectangle.rows, CV_8UC1);
    cv::Mat filtered2(rectangle.cols, rectangle.rows, CV_8UC1);
    cv::filter2D(rectangle, filtered1, CV_32F, filter1, anchor, 0, cv::BORDER_DEFAULT);
    cv::filter2D(rectangle, filtered2, CV_32F, filter2, anchor, 0, cv::BORDER_DEFAULT);

    // В отличие от первого способа фильтр sqrt(f1^2 + f2^2)
    // создается не из ядер первых двух фильтров, а из результатов применения этих фильтров
    cv::Mat filteredSqrt;
    cv::pow(filtered1.mul(filtered1) + filtered2.mul(filtered2), 0.5, filteredSqrt);

    // Нормализация
    filtered1 = ((filtered1 + 255) / 2);
    filtered2 = ((filtered2 + 255) / 2);

    // Конвертирование из float 32 в uchar 8
    filtered1.convertTo(filtered1, CV_8UC1);
    filtered2.convertTo(filtered2, CV_8UC1);
    filteredSqrt.convertTo(filteredSqrt, CV_8UC1);

    // -------------------------------------------------------------------------------------

    // Указать свой путь
    std::string writeTo{ "C:/Users/dimam/Documents/Admin/Programming/C++/OpenCV/OpenCVData/test_data/"
         };
```

```
    cv::imwrite(writeTo + "original_image.png", rectangle, { CV_IMWRITE_PNG_COMPRESSION, 0 });
    cv::imwrite(writeTo + "filtered1.png", filtered1, { CV_IMWRITE_PNG_COMPRESSION, 0 });
    cv::imwrite(writeTo + "filtered2.png", filtered2, { CV_IMWRITE_PNG_COMPRESSION, 0 });
    cv::imwrite(writeTo + "filteredSqrt.png", filteredSqrt, { CV_IMWRITE_PNG_COMPRESSION, 0 });

    return 0;
}
```