数字图像处理 作业 6

朱文杰 220320623 自动化 6 班 | 2024.10.22

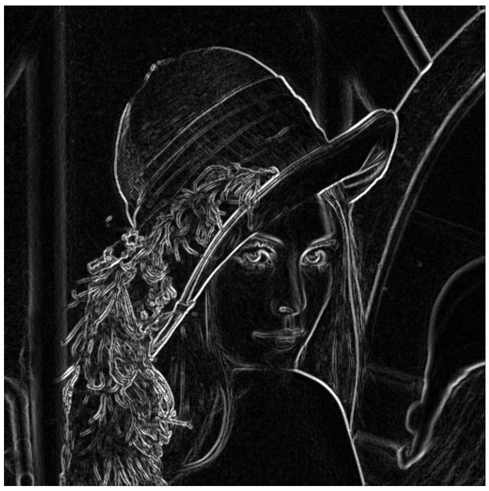
> **1** 用 MATLAB 或 OpenCV 对 Lena 的图片进行边缘提取，使用 Sobel, Roberts, Prewitt, LoG, Canny 算法。

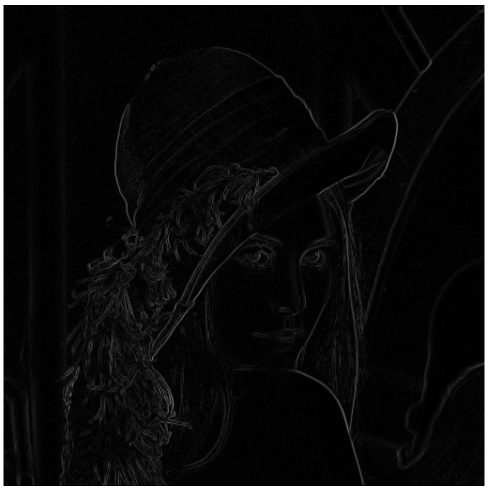
OpenCV 代码：

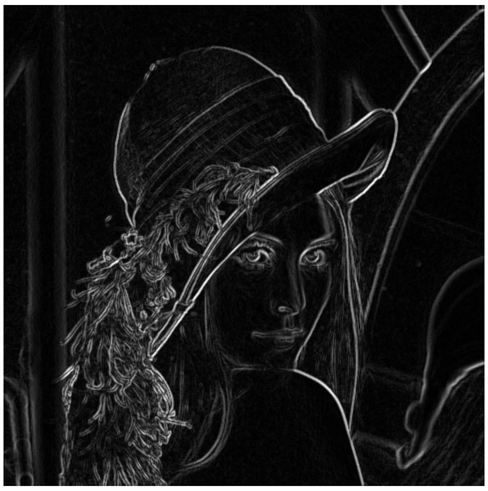
```
#include <opencv2/opencv.hpp>
#include <opencv2/highgui.hpp>
#include <filesystem>

using namespace cv;
using namespace std;
namespace fs = std::filesystem;

int main(int argc, char const *argv[])
{
    fs::path source_path = __FILE__;
    fs::path source_dir = source_path.parent_path();

    string image_path = (source_dir / "images/lena.png").string();
    Mat img = imread(image_path, IMREAD_COLOR);
    if (img.empty()) {
        cerr << "Could not open or find the image: " << image_path << endl;
        return -1;
    }
    imshow("Lena", img);

    // 用 Sobel 算子进行边缘提取
    Mat gray, grad_x, grad_y, abs_grad_x, abs_grad_y;
    cvtColor(img, gray, COLOR_BGR2GRAY);
    Sobel(gray, grad_x, CV_16S, 1, 0, 3);
    Sobel(gray, grad_y, CV_16S, 0, 1, 3);
    convertScaleAbs(grad_x, abs_grad_x);
    convertScaleAbs(grad_y, abs_grad_y);
    Mat sobel;
    addWeighted(abs_grad_x, 0.5, abs_grad_y, 0.5, 0, sobel);
    imshow("Sobel", sobel);

    // 用 Roberts 算子进行边缘提取
    Mat grad_x2, grad_y2, abs_grad_x2, abs_grad_y2;
    grad_x2 = (Mat_<int>(2, 2) << 1, 0, 0, -1);
    grad_y2 = (Mat_<int>(2, 2) << 0, 1, -1, 0);
    filter2D(gray, abs_grad_x2, CV_16S, grad_x2);
    filter2D(gray, abs_grad_y2, CV_16S, grad_y2);
    convertScaleAbs(abs_grad_x2, abs_grad_x2);
    convertScaleAbs(abs_grad_y2, abs_grad_y2);
    Mat roberts;
    addWeighted(abs_grad_x2, 0.5, abs_grad_y2, 0.5, 0, roberts);
    imshow("Roberts", roberts);

    // 用 Prewitt 算子进行边缘提取
    Mat kernel_x = (Mat_<int>(3, 3) << -1, 0, 1, -1, 0, 1, -1, 0, 1);
    Mat kernel_y = (Mat_<int>(3, 3) << -1, -1, -1, 0, 0, 0, 1, 1, 1);
    Mat grad_x3, grad_y3, abs_grad_x3, abs_grad_y3;
    filter2D(gray, grad_x3, CV_16S, kernel_x);
    filter2D(gray, grad_y3, CV_16S, kernel_y);
    convertScaleAbs(grad_x3, abs_grad_x3);
    convertScaleAbs(grad_y3, abs_grad_y3);
    Mat prewitt;
    addWeighted(abs_grad_x3, 0.5, abs_grad_y3, 0.5, 0, prewitt);
    imshow("Prewitt", prewitt);

    // 用 LoG 算子进行边缘提取
    Mat laplacian;
    Laplacian(gray, laplacian, CV_16S, 3);
    convertScaleAbs(laplacian, laplacian);
    imshow("LoG", laplacian);

    // 用 Canny 算子进行边缘提取
    Mat blur;
    GaussianBlur(gray, blur, Size(3, 3), 0, 0); // 先进行高斯模糊
    Mat canny;
    Canny(blur, canny, 50, 150);
    imshow("Canny", canny);

    waitKey(0);
    return 0;
}
```

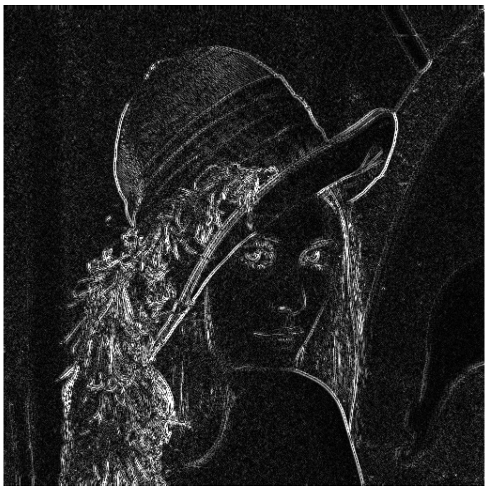
运行结果：

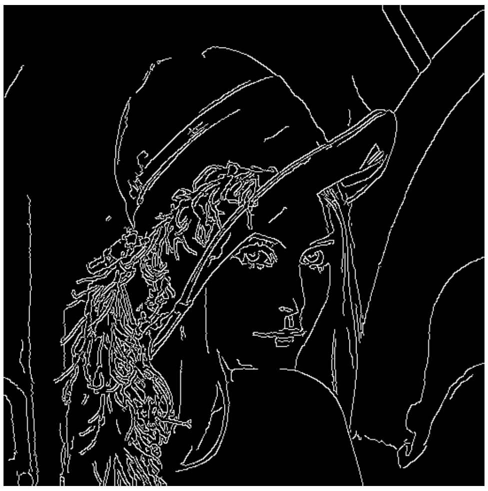
Lena


Sobel


Roberts


Prewitt


LoG

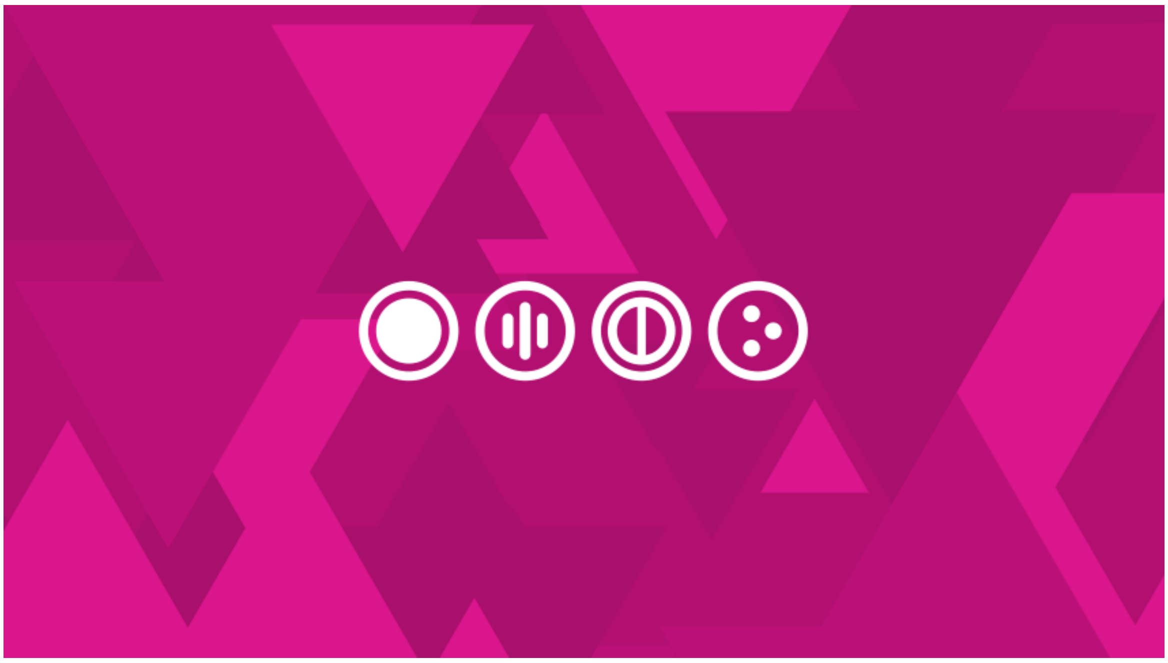

Canny

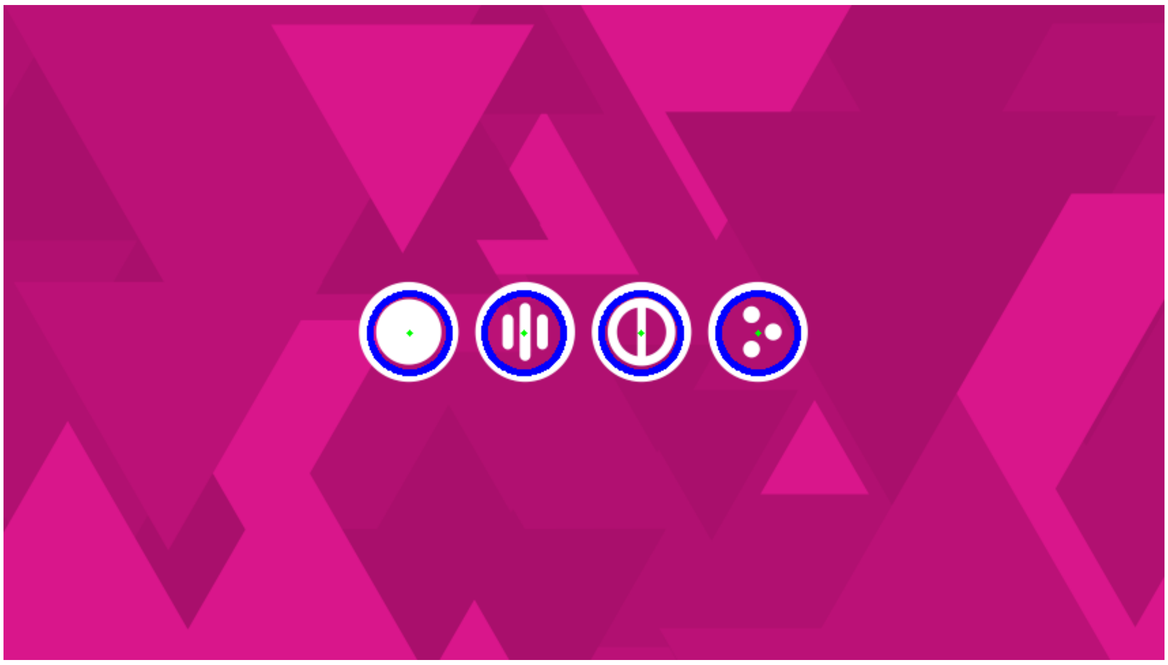
> **2** 参考 https://www.cnblogs.com/bjxqmy/p/12333022.html 的方法实现基于 Hough Transform 的圆检测。

OpenCV 代码：

```
#include <opencv2/opencv.hpp>
#include <opencv2/highgui.hpp>
#include <filesystem>

using namespace cv;
using namespace std;
namespace fs = std::filesystem;

void hough_circle(Mat &gray, Mat &result, int hough_threshold);

int main(int argc, char const *argv[])
{
    fs::path source_path = __FILE__;
    fs::path source_dir = source_path.parent_path();
    string image_path = (source_dir / "images/osu.png").string();
    Mat img = imread(image_path, IMREAD_COLOR);
    if (img.empty()) {
        cerr << "Could not open or find the image: " << image_path << endl;
        return -1;
    }
    imshow("Original", img);

    Mat gray, result;
    cvtColor(img, gray, COLOR_BGR2GRAY);
    result = img.clone();

    int hough_threshold = 100;
    namedWindow("Hough Circle", WINDOW_AUTOSIZE);
    hough_circle(gray, result, hough_threshold);
    imshow("Hough Circle", result);

    waitKey(0);
    return 0;
}



void hough_circle(Mat &gray, Mat &result, int hough_threshold)
{
    vector<Vec3f> circles;
    HoughCircles(gray, circles, HOUGH_GRADIENT, 1, 10, hough_threshold,
hough_threshold / 2);

    for (int i = 0; i < circles.size(); i++)
    {
        Point center(cvRound(circles[i][0]), cvRound(circles[i][1]));
        int radius = cvRound(circles[i][2]);
        circle(result, center, 2, Scalar(0, 255, 0), -1, 8, 0);
        circle(result, center, radius, Scalar(255, 0, 0), 3, 8, 0);
    }
}
```

运行结果:


Original


Hough Circle