Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

Дисциплина: Методы и средства цифровой обработки информации

Выполнили студенты гр. 43501/1 А.А. Дроздовский
Руководитель Н.А. Абрамов

«___» _____ 2017г.

Санкт -Петербург
2017

СОДЕРЖАНИЕ

1. Цель работы

Изучить операторы свертки изображения используемые для нахождения границ на изображение.

2. Ход работы

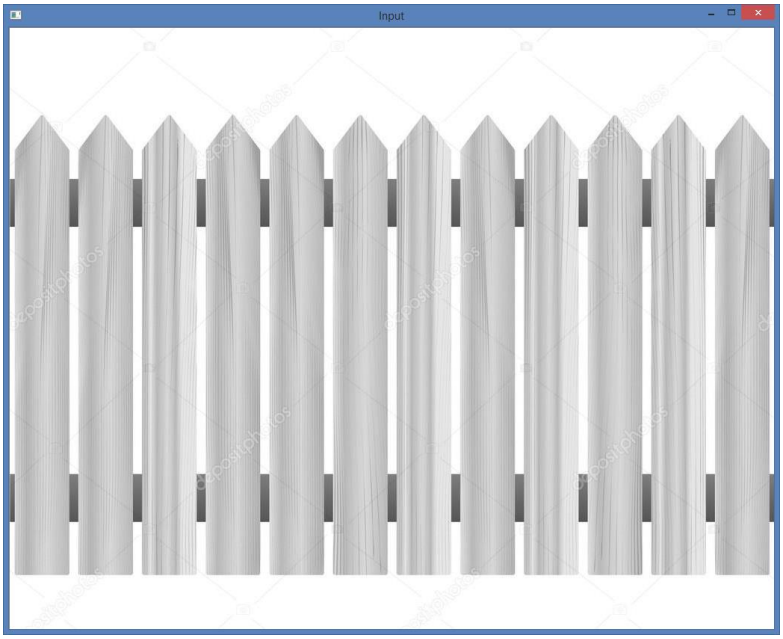

Рис. 2.1. Исходное изображение 1.

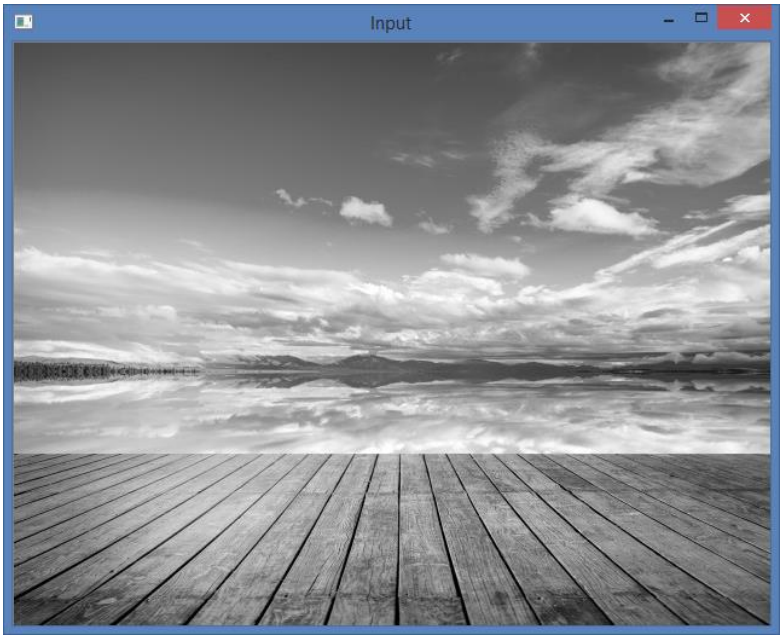

Рис. 2.2. Исходное изображение 2.

Рис. 2.3. Исходное изображение 3.

2.1. Оператор Собеля

$$K_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix};$$

$$K_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix};$$

$G_x = K_x \cdot Image;$

$G_y = K_y \cdot Image$, где

Image – Исходное изображение;

G_x, G_y – два изображения, на которых каждая точка содержит приближённые производные по *X* и по *Y*.

$G = \sqrt{G_x^2 + G_y^2};$

G – Выходное изображение.

- Вертикальные линии

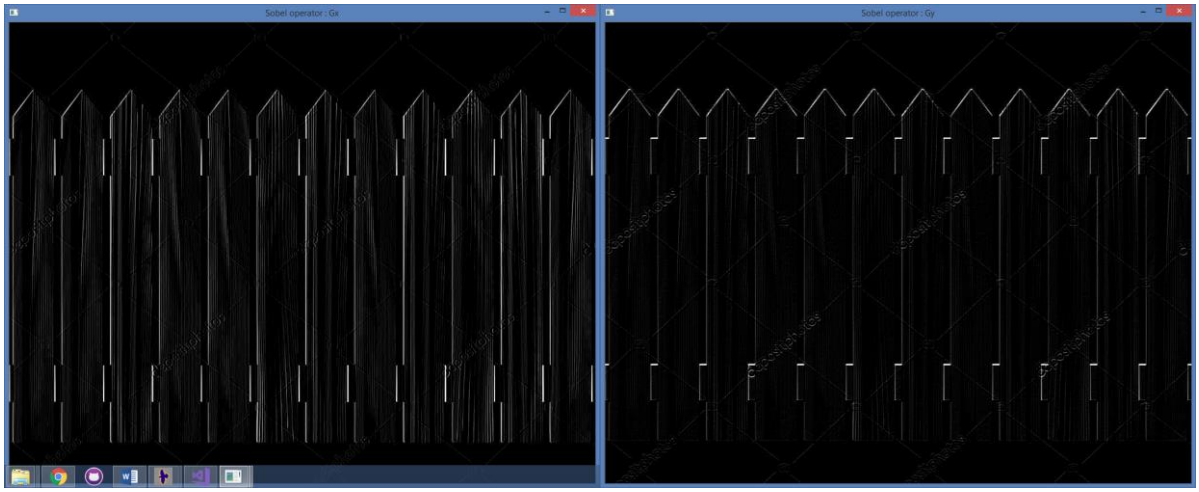

Рис. 2.4. Полученные Gx и Gy изображения 1 с помощью оператора Собеля.

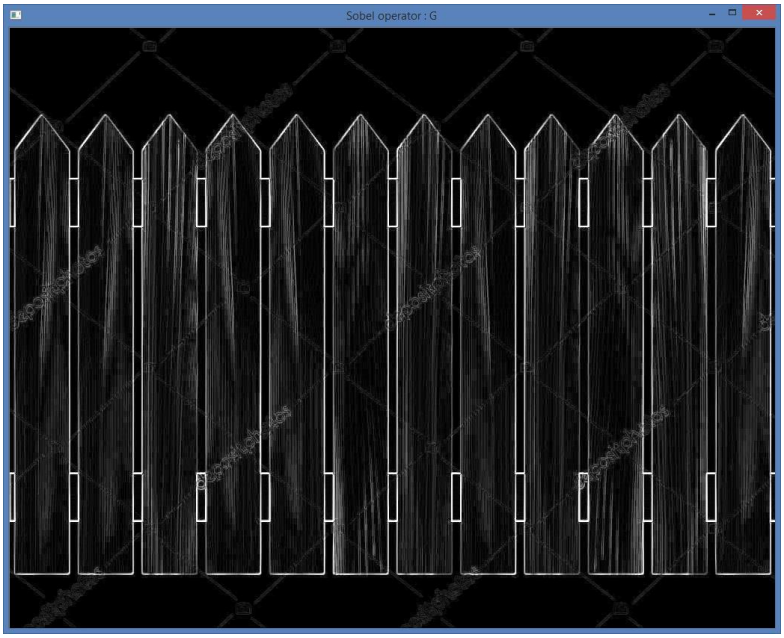

Рис. 2.5. Выходное изображение 1 с помощью оператора Собеля.

- Горизонтальные линии

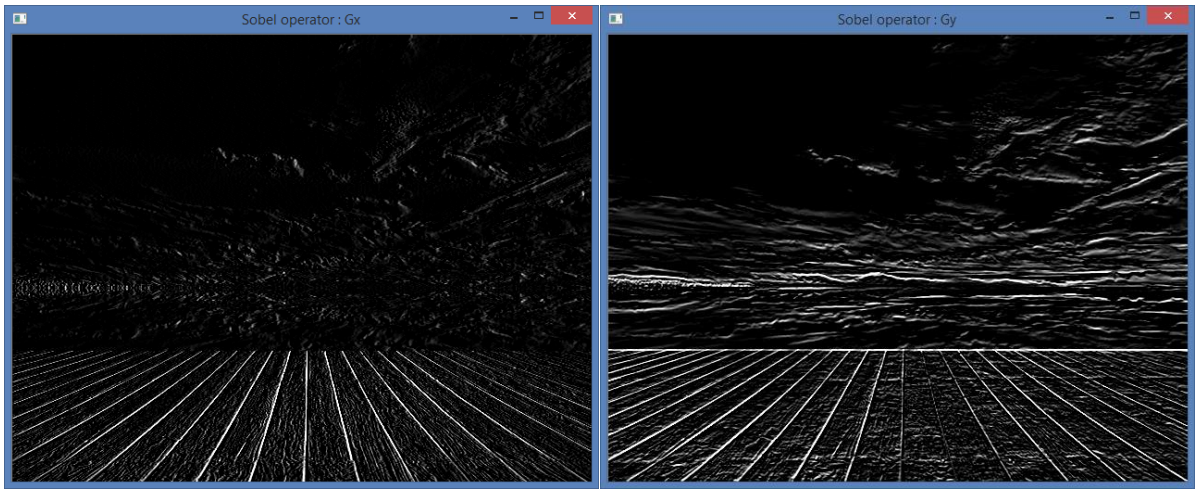

Рис. 2.6. Полученные Gx и Gy изображения 2 с помощью оператора Собеля.

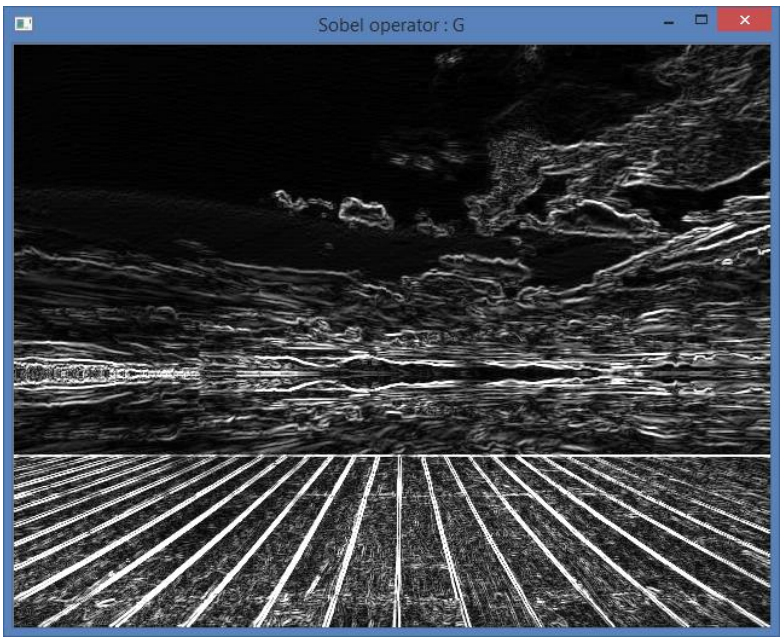

Рис. 2.7. Выходное изображение 2 с помощью оператора Собеля.

- Диагональные линии

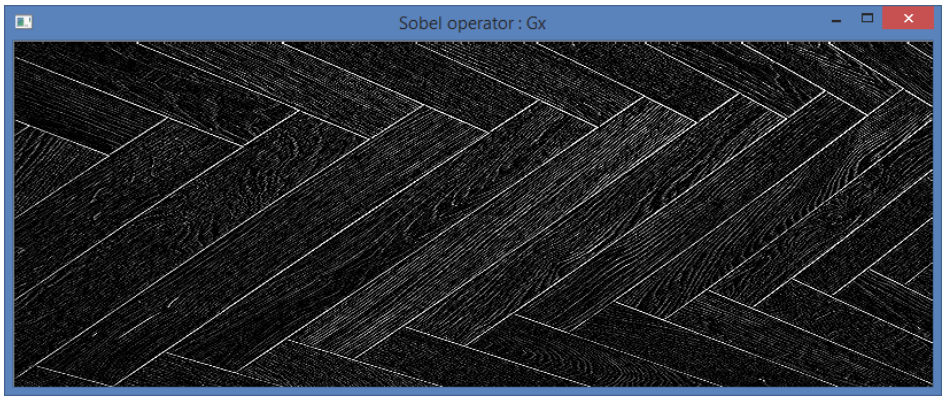

Рис. 2.8. Полученные Gx изображения 3 с помощью оператора Собеля.

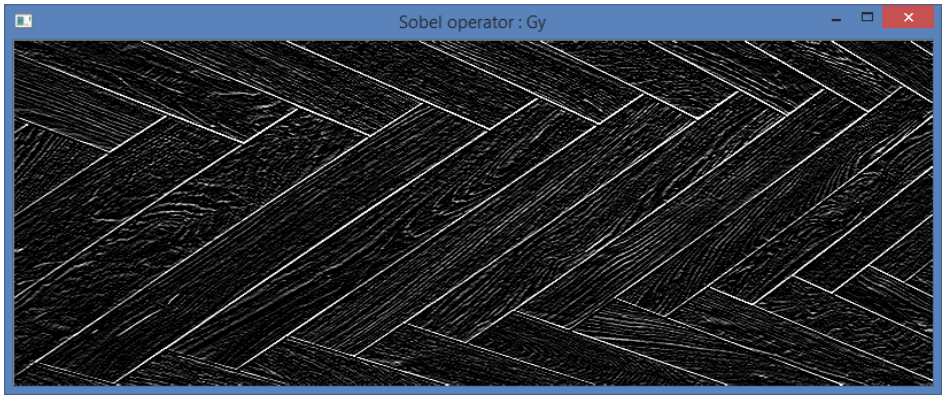

Рис. 2.9. Полученные Gy изображения 3 с помощью оператора Собеля.

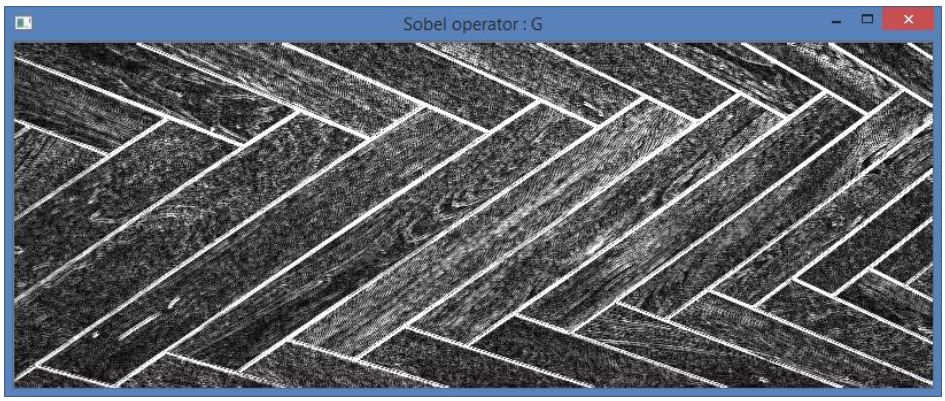

Рис. 2.10. Выходное изображение 3 с помощью оператора Собеля.

2.2. Оператор Прюитт

$$K_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix};$$

$$K_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix};$$

- Вертикальные линии

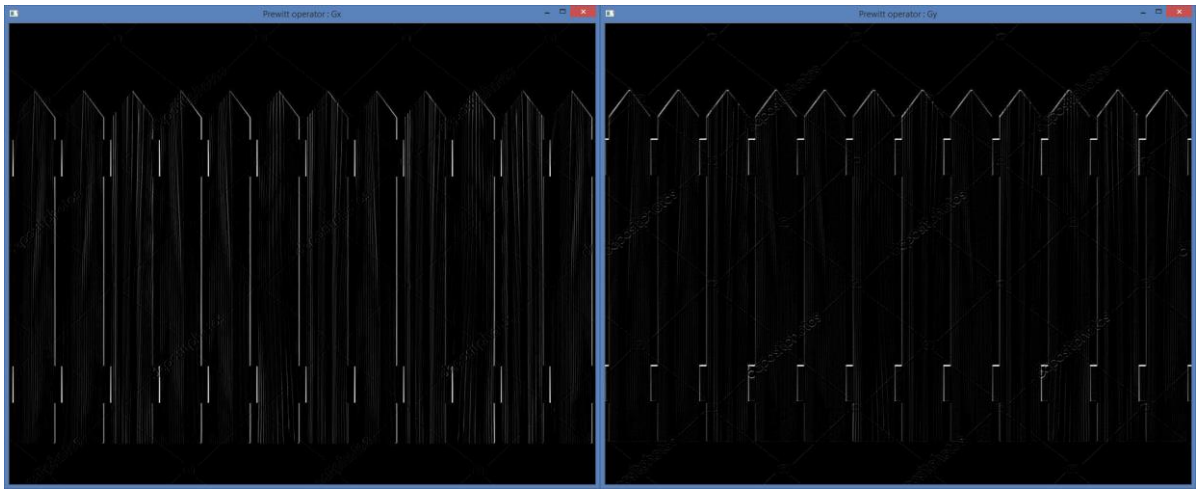

Рис. 2.11. Полученные Gx Gy изображения 1 с помощью оператора Приютт.

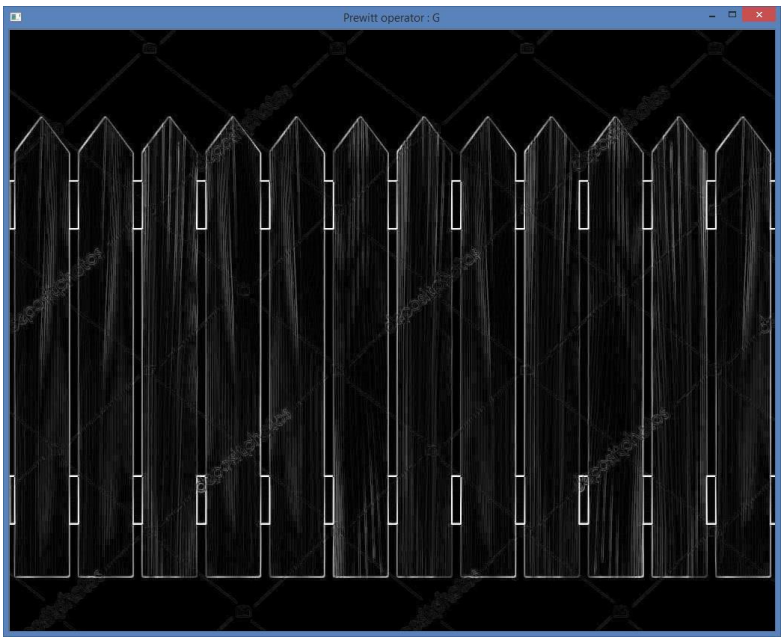

Рис. 2.12. Выходное изображение 1 с помощью оператора Приютт.

- Горизонтальные линии

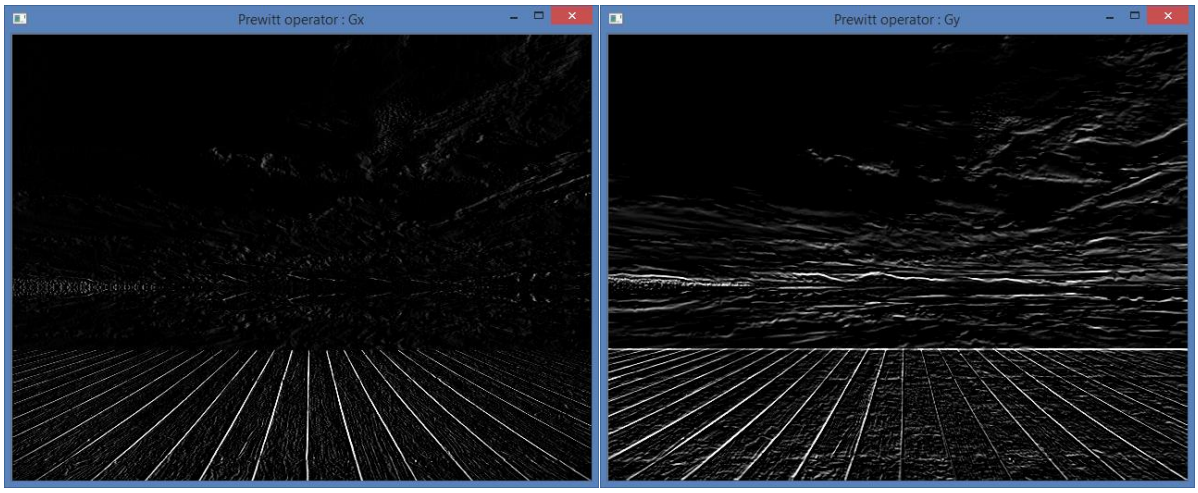

Рис. 2.13. Полученные Gx Gy изображения 2 с помощью оператора Приютт.

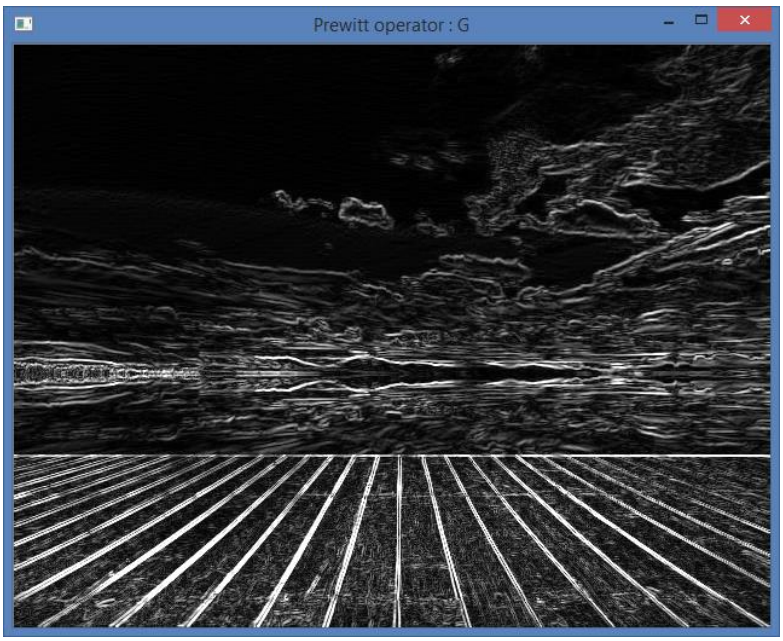

Рис. 2.14. Выходное изображение 2 с помощью оператора Приютт.

- Диагональные линии

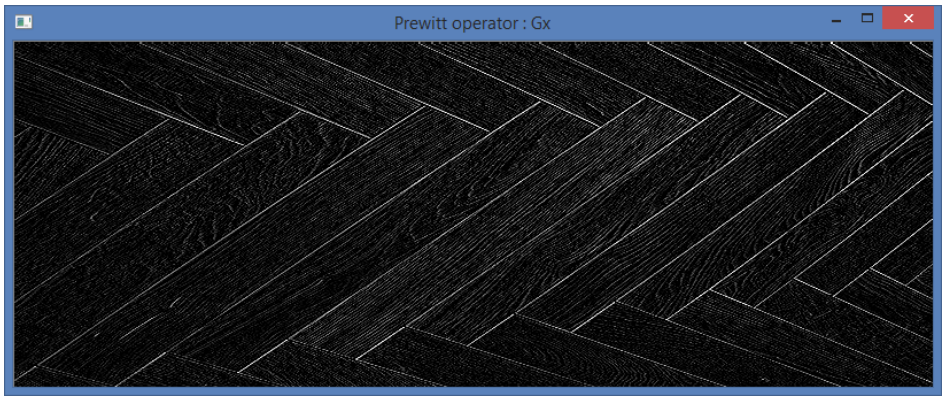

Рис. 2.15. Полученные Gx изображения 3 с помощью оператора Приютт.

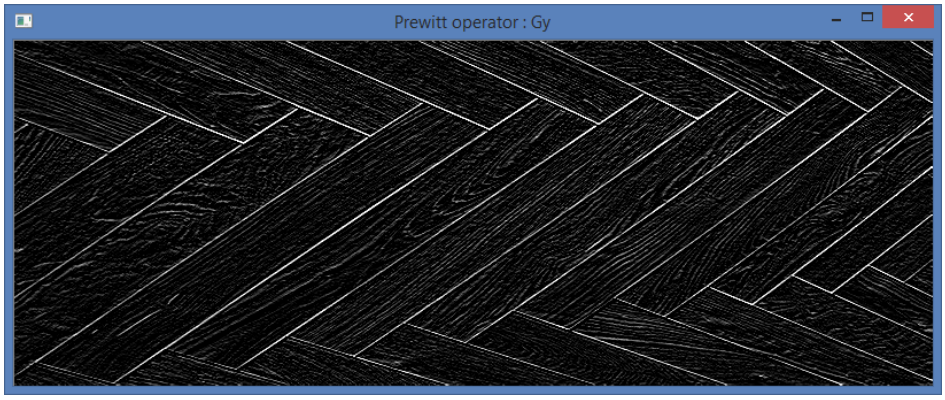

Рис. 2.16. Полученные Gy изображения 3 с помощью оператора Приютт.

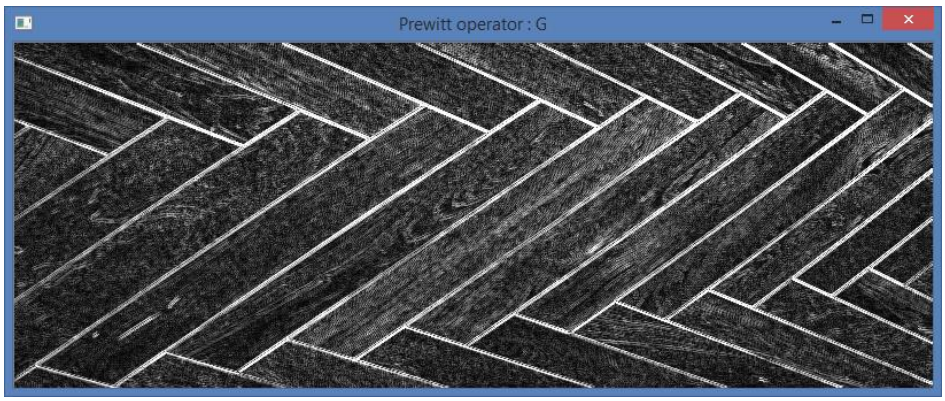

Рис. 2.17. Выходное изображение 3 с помощью оператора Приютт.

2.3. Перекрёстный оператор Робертса

$$K_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix};$$

$$K_y = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix};$$

- Вертикальные линии

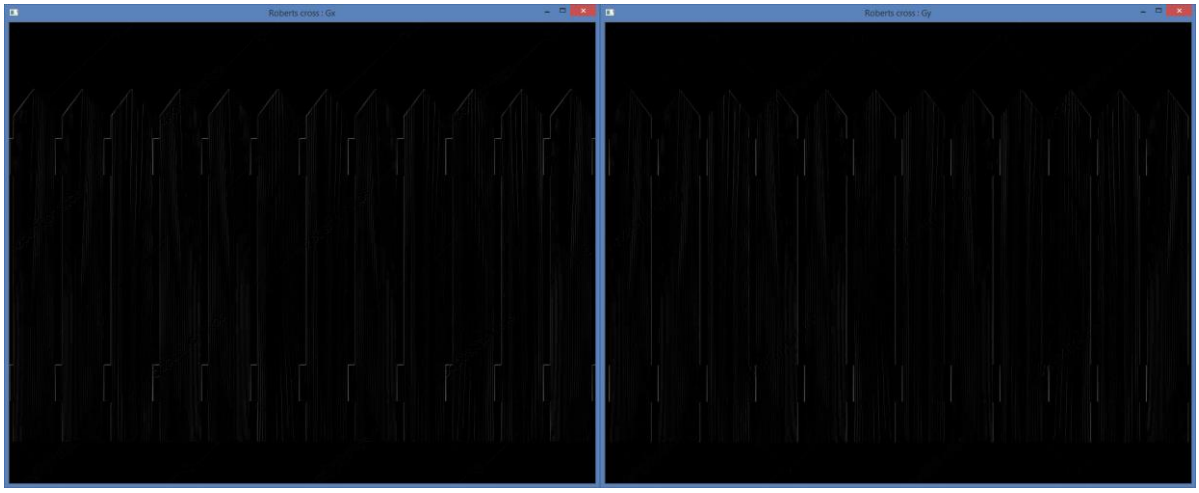

Рис. 2.18. Полученные Gx Gy изображения 1 с помощью оператора Робертса.

Рис. 2.19. Выходное изображение 1 с помощью оператора Робертса.

- Горизонтальные линии

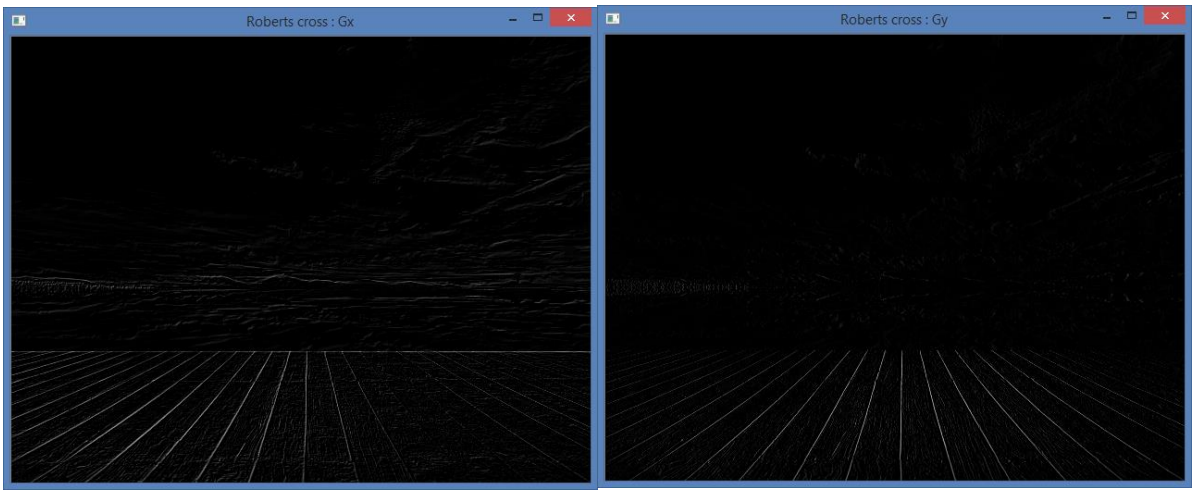

Рис. 2.20. Полученные Gx Gy изображения 2 с помощью оператора Робертса.

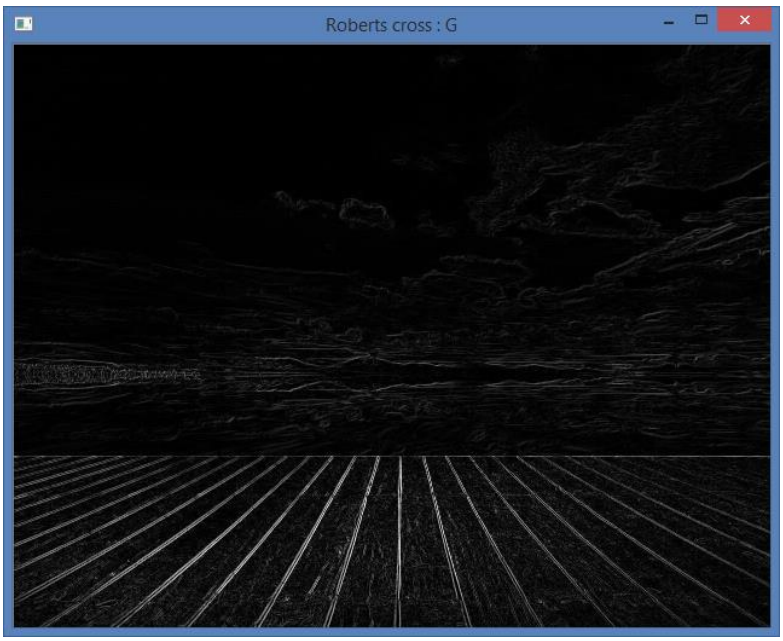

Рис. 2.21. Выходное изображение 2 с помощью оператора Робертса.

- Диагональные линии

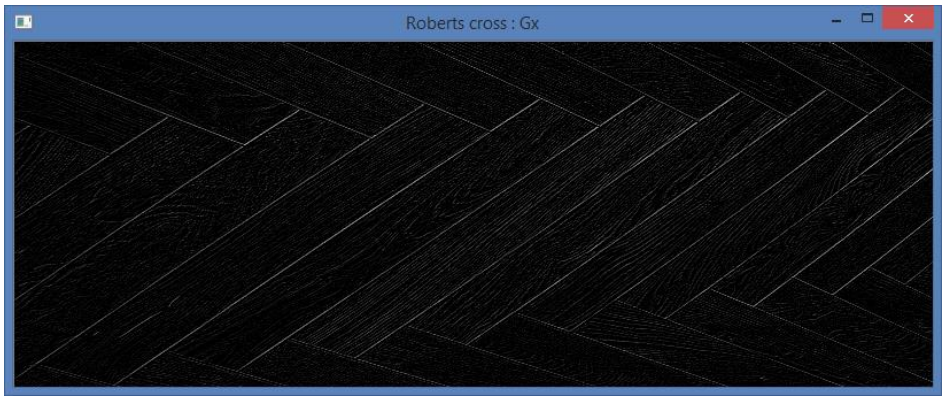

Рис. 2.22. Полученные Gx изображения 3 с помощью оператора Робертса.

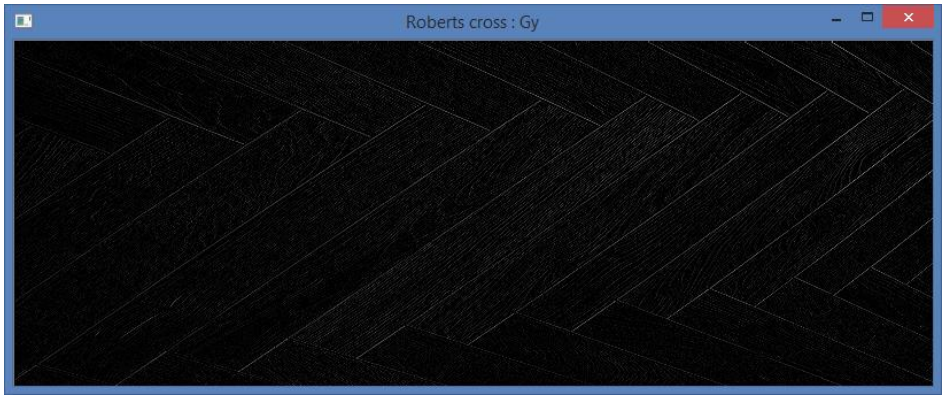

Рис. 2.23. Полученные Gy изображения 3 с помощью оператора Робертса.

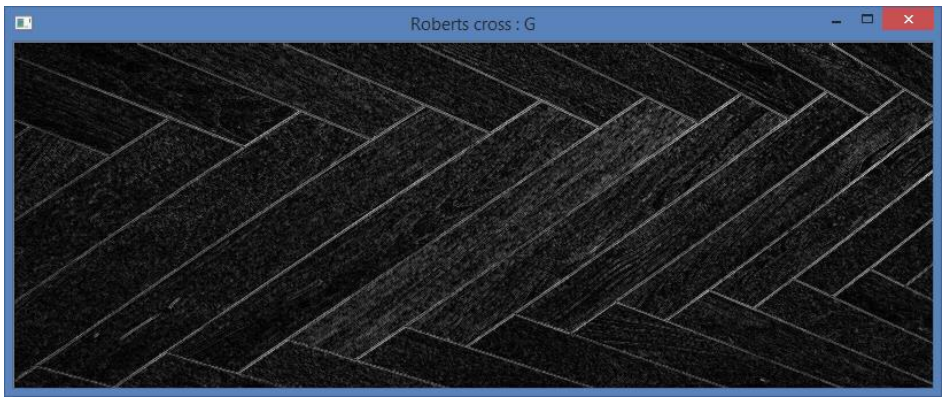

Рис. 2.24. Выходное изображение 3 с помощью оператора Робертса.

3. Выводы

Список используемой литературы

1. Sobel operator [Электронный ресурс] // Википедия : свободная энцикл., 2017. – URL: https://en.wikipedia.org/wiki/Sobel_operator (Дата обращения: 05.10.2017).
2. Prewitt operator [Электронный ресурс] // Википедия : свободная энцикл., 2017. – URL: https://en.wikipedia.org/wiki/Prewitt_operator (Дата обращения: 07.10.2017).
3. Roberts cross [Электронный ресурс] // Википедия : свободная энцикл., 2017. – URL: https://en.wikipedia.org/wiki/Roberts_cross (Дата обращения: 07.10.2017).

Текст программы

```
// LR2.cpp: определяет точку входа для консольного приложения.
//

#include "stdafx.h"
#include <opencv2\opencv.hpp>
#include <math.h>
#include <iostream>
#include <string>

using namespace cv;
using namespace std;

#define To_Float static_cast<float>

void Matrix_multiplication(Mat &G, Mat image, int K[9]);
void Matrix_addition(Mat &G, Mat Gx, Mat Gy);
void Delete(Mat &m);
void Formulation(Mat Gx, Mat Gy, Mat G, String S);

int main(int argc, char** argv)
{
	int Sobel_operator_x[9] = { 1,0,-1,2,0,-2,1,0,-1 };
	int Sobel_operator_y[9] = { 1,2,1,0,0,0,-1,-2,-1 };
	int Prewitt_operator_x[9] = { -1,0,1,-1,0,1,-1,0,1 };
	int Prewitt_operator_y[9] = { 1,1,1,0,0,0,-1,-1,-1 };
	int Roberts_operator_x[9] = { 1,0,0,0,-1,0,0,0,0 };
	int Roberts_operator_y[9] = { 0,1,0,-1,0,0,0,0,0 };

	namedWindow("LR2");
	String imageName("LR2_3.jpg");
	if (argc > 1)
	{
		imageName = argv[1];
	}
	/// Считывание изображения в оттенках серого
	Mat image = imread(imageName.c_str(), IMREAD_GRAYSCALE);
	Mat image_with_frame(image.rows+2, image.cols+2, CV_8UC1, Scalar(255, 255,
255)); // CV_8UC1 - 2^8 uchar = 0 до 255
	Mat Gx(image.rows, image.cols, CV_16SC1, Scalar(255, 255, 255));
			// CV_16SC1 -  2^16 short = -32 768 до 32 767
	Mat Gy(image.rows, image.cols, CV_16SC1, Scalar(255, 255, 255));
	Mat G(image.rows, image.cols, CV_16SC1, Scalar(255, 255, 255));
	/// Вывод исходного изображения
	imshow("Input", image);

	/// Копия изображения с белой рамкой
	for (int i = 1; i <= image.rows; i++)
		for (int j = 1; j <= image.cols; j++)
			image_with_frame.at<uchar>(i, j) = image.at<uchar>(i-1, j-1);
	/// Заполнение белой рамки соседними цветами
	// Заполнение вехний и нижний границы
	for (int j = 0; j < image_with_frame.cols; j++)
	{
		image_with_frame.at<uchar>(0, j) = image_with_frame.at<uchar>(1, j);
		image_with_frame.at<uchar>(image_with_frame.rows-1, j) =
image_with_frame.at<uchar>(image_with_frame.rows-2, j);
	}
	// Заполнение левой и правой границы
	for (int i = 0; i < image_with_frame.rows; i++)
	{
```

```
            image_with_frame.at<uchar>(i, 0) = image_with_frame.at<uchar>(i, 1);
            image_with_frame.at<uchar>(i, image_with_frame.cols - 1) =
image_with_frame.at<uchar>(i, image_with_frame.cols - 2);
      }
      /*-----------=== Часть 1 (Оператор Собеля) ===-----------*/
      Matrix_multiplication(Gx, image_with_frame, Sobel_operator_x);
      Matrix_multiplication(Gy, image_with_frame, Sobel_operator_y);
      Formulation(Gx, Gy, G, "Sobel operator");
      /// Обнуление переменных
      Delete(Gx);
      Delete(Gy);
      Delete(G);

      /*-----------=== Часть 2 (Оператор Прюитт) ===-----------*/
      Matrix_multiplication(Gx, image_with_frame, Prewitt_operator_x);
      Matrix_multiplication(Gy, image_with_frame, Prewitt_operator_y);
      Formulation(Gx, Gy, G, "Prewitt operator");
      /// Обнуление переменных
      Delete(Gx);
      Delete(Gy);
      Delete(G);

      /*-----------=== Часть 3 (Оператор Робертса) ===-----------*/
      Matrix_multiplication(Gx, image_with_frame, Roberts_operator_x);
      Matrix_multiplication(Gy, image_with_frame, Roberts_operator_y);
      Formulation(Gx, Gy, G, "Roberts cross");

      waitKey(0);
   return 0;
}

///    [k0 k1 k2]
///    [k3 k4 k5]
///    [k6 k7 k8]
void Matrix_multiplication(Mat &G, Mat image, int K[9])
{
      for (int i = 1; i < image.rows - 1; i++)
            for (int j = 1; j < image.cols - 1; j++)
                  G.at<short>(i - 1, j - 1) =
                  K[0] * To_Float(image.at<uchar>(i - 1, j - 1)) + K[1] *
To_Float(image.at<uchar>(i - 1, j)) + K[2] * To_Float(image.at<uchar>(i - 1, j + 1))
                  + K[3] * To_Float(image.at<uchar>(i - 1, j - 1)) + K[4] *
To_Float(image.at<uchar>(i, j)) + K[5] * To_Float(image.at<uchar>(i - 1, j + 1))
                  + K[6] * To_Float(image.at<uchar>(i + 1, j + 1)) + K[7] *
To_Float(image.at<uchar>(i + 1, j)) + K[8] * To_Float(image.at<uchar>(i + 1, j + 1));
}
/// Умножение матриц
void Matrix_addition(Mat &G, Mat Gx, Mat Gy)
{
      for (int i = 0; i < G.rows * G.cols; i++)
            G.at<short>(i) =
sqrt((To_Float(Gx.at<short>(i)))*(To_Float(Gx.at<short>(i)))
                  + (To_Float(Gy.at<short>(i)))*(To_Float(Gy.at<short>(i))));
}

/// Обнуление Mat (Белый фон)
void Delete(Mat &m)
{
      for (int i = 0; i < m.cols*m.rows; i++)
            switch (m.type())
            {
            case CV_8UC1: m.at<uchar>(i) = 255;
                  break;
            case CV_8SC1: m.at<char>(i) = 127;
```

```
                        break;
                case CV_16UC1:      m.at<ushort>(i) = 65535;
                        break;
                case CV_16SC1:      m.at<short>(i) = 32767;
                        break;
                default:
                        break;
                }
}

/// Формализация
void Formulation(Mat Gx, Mat Gy, Mat G, String S)
{
        Matrix_addition(G, Gx, Gy);
        Mat G_uchar(G.rows, G.cols, CV_8UC1, Scalar(255, 255, 255));
        Mat Gx_uchar(Gx.rows, Gx.cols, CV_8UC1, Scalar(255, 255, 255));
        Mat Gy_uchar(Gy.rows, Gy.cols, CV_8UC1, Scalar(255, 255, 255));
        Gx.convertTo(Gx_uchar, CV_8UC1);
        Gy.convertTo(Gy_uchar, CV_8UC1);
        G.convertTo(G_uchar, CV_8UC1);
        /// Вывод полученных изображений на экран
        imshow(S + " : Gx", Gx_uchar);
        imshow(S + " : Gy", Gy_uchar);
        imshow(S + " : G", G_uchar);
}
```