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

Отчёт о лабораторной работе №3

Дисциплина: Методы и средства цифровой обработки информации

Выполнил студент гр. 13541/1 _____ А.А. Дроздовский
(подпись)
Руководитель _____ Н.А. Абрамов
(подпись)
«__» _____ 2017 г.

Санкт – Петербург
2017

СОДЕРЖАНИЕ

1. Цель работы

Изучить методы нахождения границ на изображении с помощью оператора Лапласиана;

изучить метод сглаживания изображения на примере фильтра Гаусса;

изучить метод повышения резкости изображения.

2. Ход работы

2.1. Фильтр Гаусса

$$G = \begin{bmatrix} 1/16 & 2/16 & 1/16 \\ 2/16 & 4/16 & 2/16 \\ 1/16 & 2/16 & 1/16 \end{bmatrix};$$

$Image_{out} = IMG_{in} \cdot G$, где

$Image_{out}$ – выходное изображение после обработки;

$Image_{in}$ – исходное изображение;

G – матрица свертки.

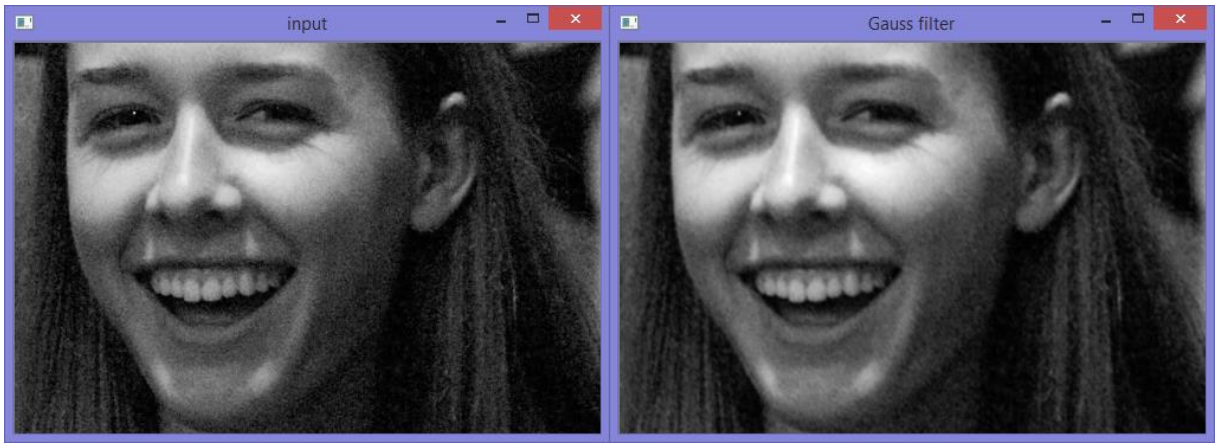

Рис. 2.1. Результат работы фильтра Гаусса. (Изображение 1)

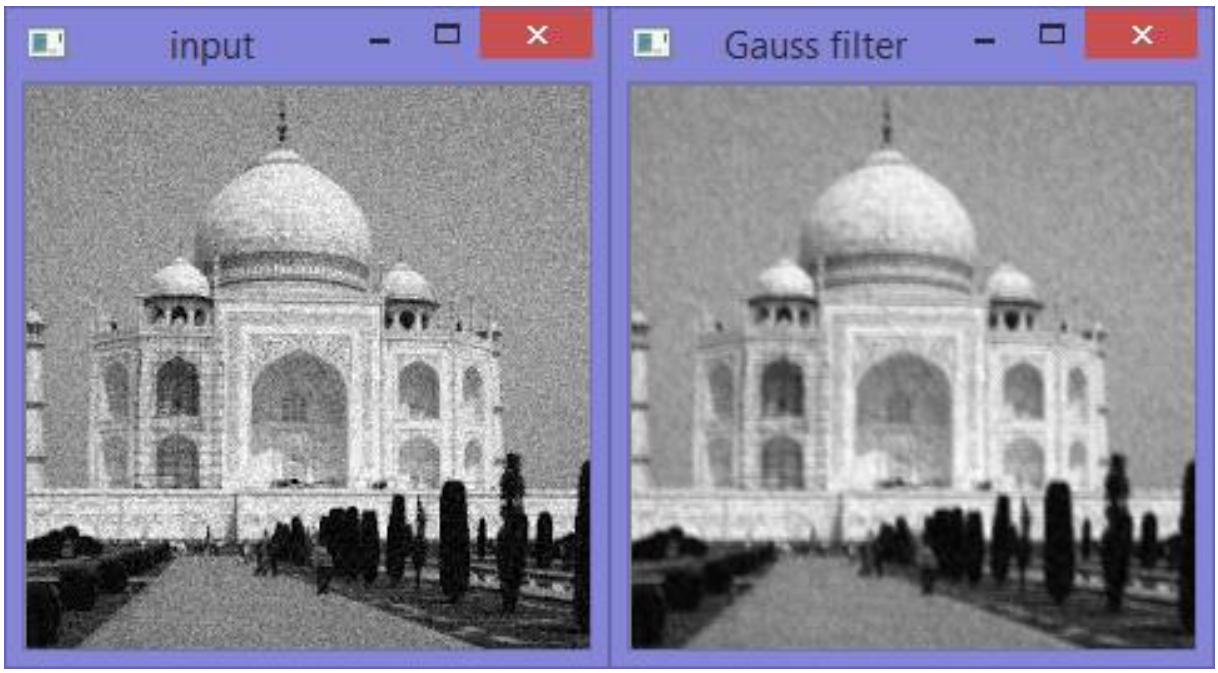

Рис. 2.2. Результат работы фильтра Гаусса. (Изображение 2)

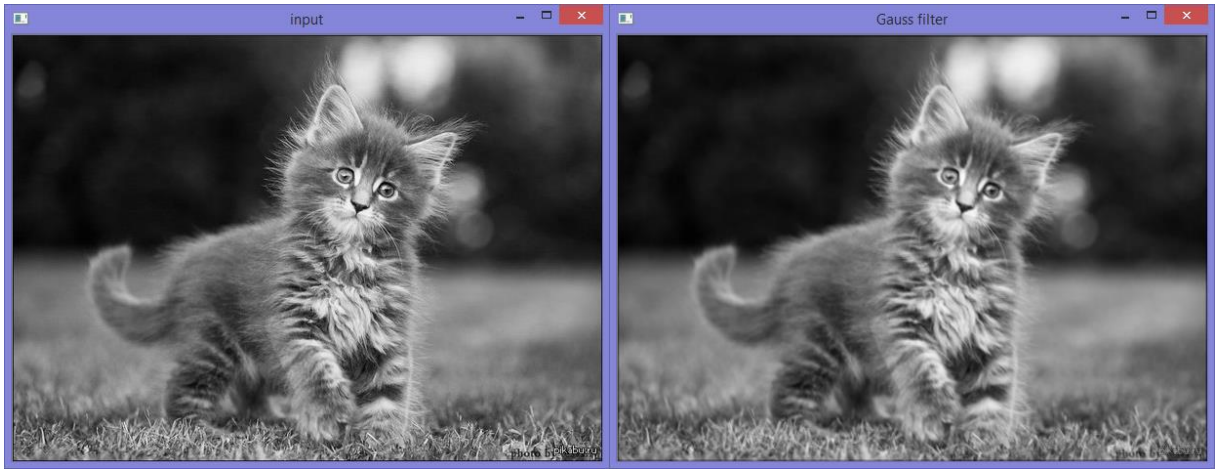

Рис. 2.3. Результат работы фильтра Гаусса. (Изображение 3)

2.2. Оператор Лапласиана

$$L = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix};$$

$Image_{out} = IMG_{in} \cdot L$, где

$Image_{out}$ – выходное изображение после обработки;

$Image_{in}$ – исходное изображение после сглаживания;

L – матрица свертки.

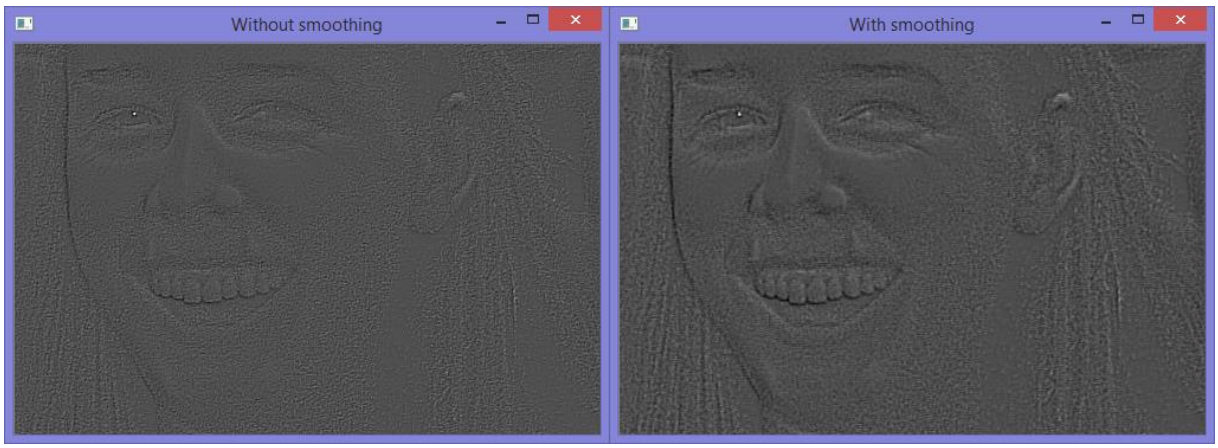

Рис. 2.4. Результат работы оператора Лапласиана. (Изображение 1)

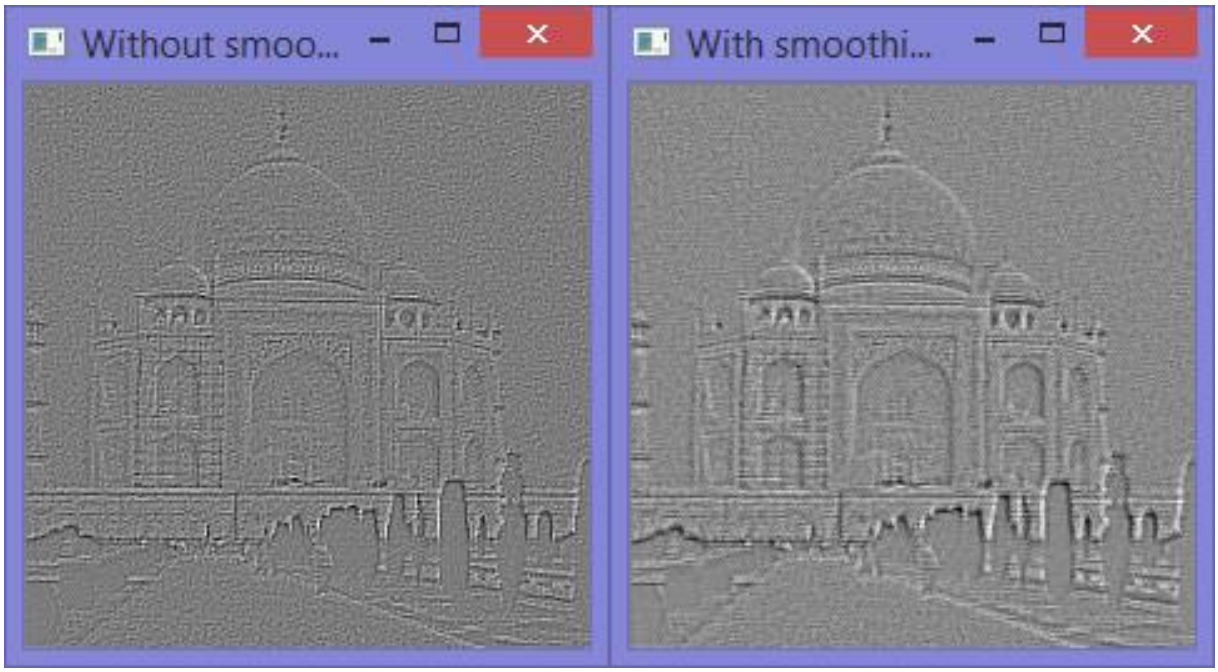

Рис. 2.5. Результат работы оператора Лапласиана. (Изображение 2)

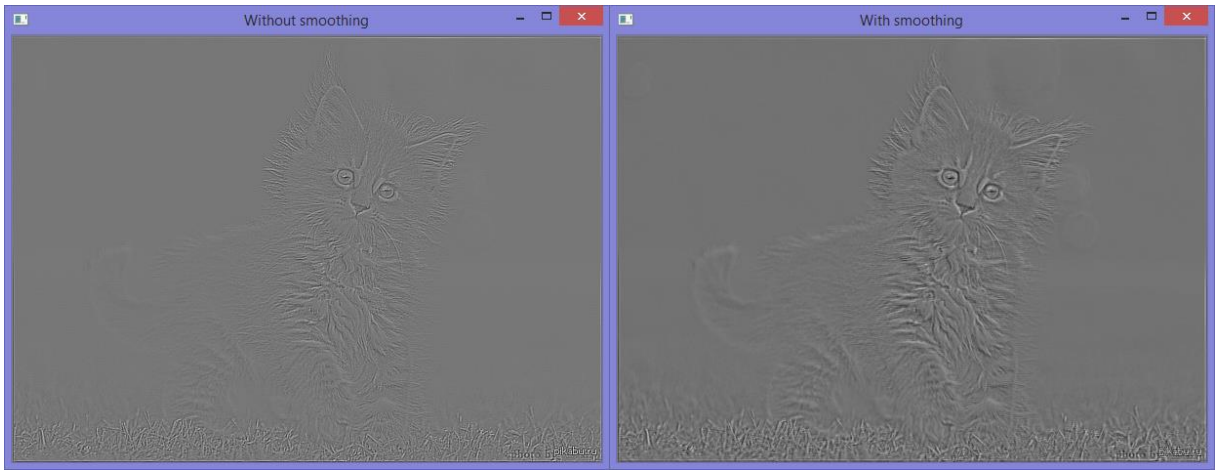

Рис. 2.5. Результат работы оператора Лапласиана. (Изображение 3)

Ниже представлено сравнение изображений с найденным границами, с использованием различных операторов. Для удобства сравнения, матрица, полученная с применением оператора Лапласиана была преобразована к виду $G = \sqrt{G^2+G^2}$:

Рис. 2.6. Лапласиан.

Рис. 2.7.Робертс.

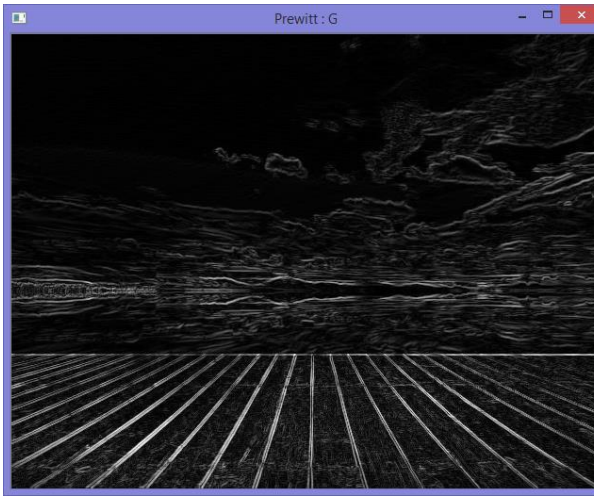

Рис. 2.8. Прюитт.

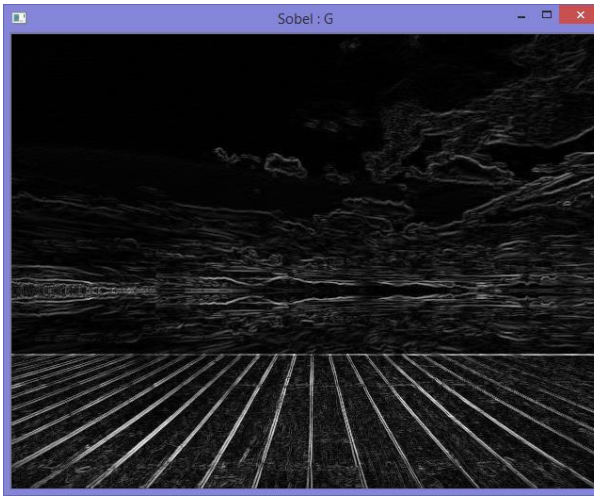

Рис. 2.9. Собель.

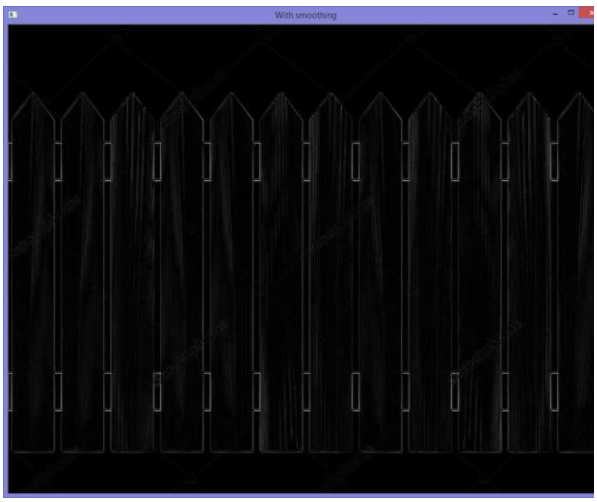

Рис. 2.10. Лапласиан.

Рис. 2.11. Робертс.

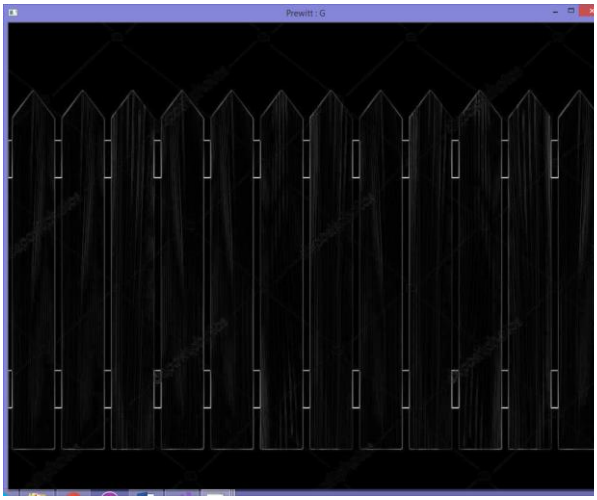

Рис. 2.12. Прюитт.

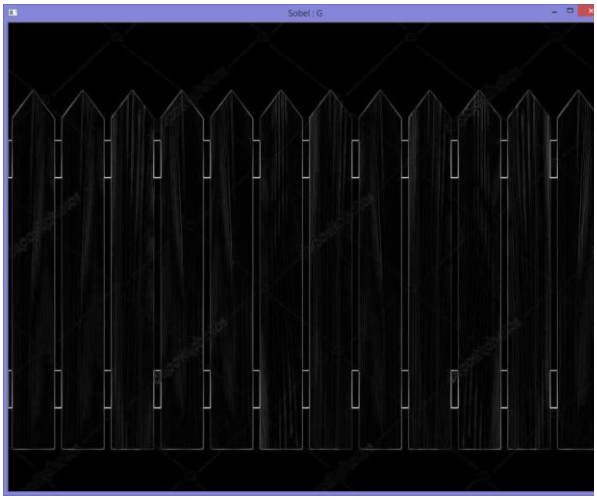

Рис. 2.13. Собель.

2.3. Повышение резкости

Для повышения резкости на 30% матрицу изображения с границами нормализовать в диапазон от 1 до 1,3. Для нахождения границ подходит любой из ранее изученных операторов. В данной работе применяются оператор Лапласиана и оператор Прюитта.

$IMG_{out} = IMG_{in} \cdot LINE_{norm}$, где

IMG_{out}, IMG_{in} – выходное и исходное изображения;

$LINE_{norm}$ – нормализованная матрица границ.

Рис. 2.14. Размытое изображение 1.

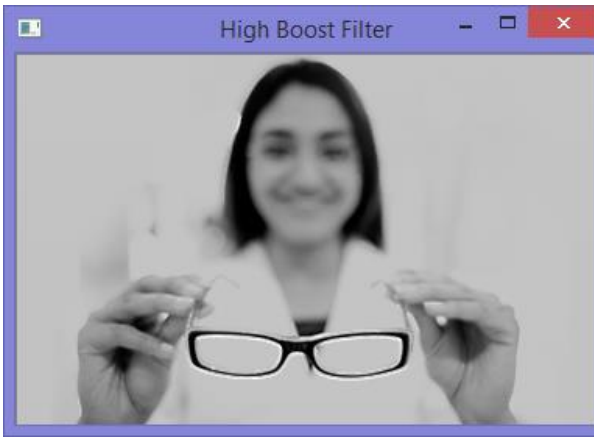

Рис. 2.15. Повышение резкости с помощью оператора Лапласиана.

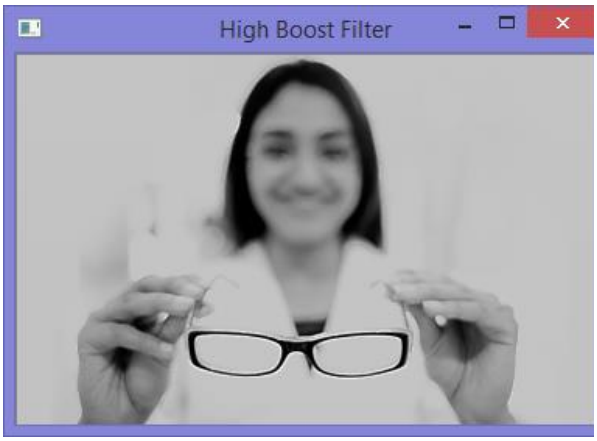

Рис. 2.16. Повышение резкости с помощью оператора Прюитт.

Ниже представлены изображения с границами, из которых были полученны изображения на рисунках 2.15 и 2.16.

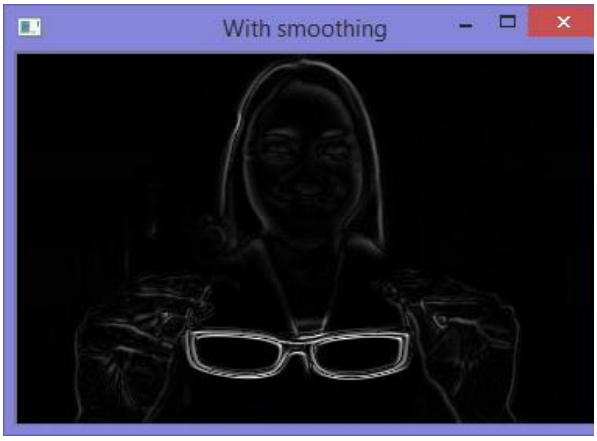

Рис. 2.17. Границы найденные с использованием Лапласиана.

Рис. 2.18. Границы найденные с использованием Прюитт.

Рис. 2.19. Размытое изображение 2.

Рис. 2.20. Повышение резкости с помощью оператора Лапласиана.

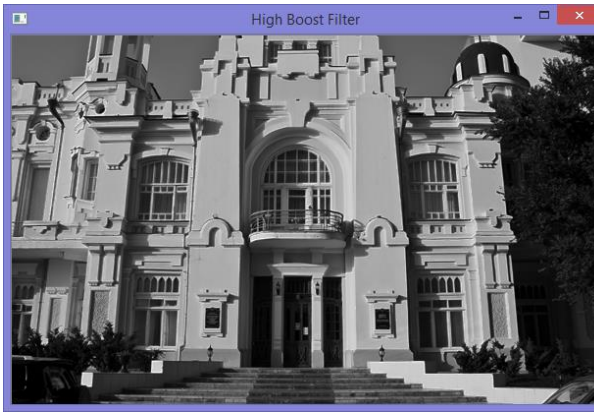

Рис. 2.21. Повышение резкости с помощью оператора Прюитт.

Рис. 2.22. Границы найденные с использованием Лапласиана.

Рис. 2.23. Границы найденные с использованием Прюитт.

3. Выводы

Сглаживающие фильтры используются в основном для снижения шума. Чем больше окно фильтрации, тем меньше средняя интенсивность шума. Побочным эффектом фильтра является размытие деталей

изображения. Матрица, которую использует фильтр Гаусса, обеспечивает более эффективное снижение шума, т.к. влияние каждого пикселя уменьшается с расстоянием.

Оператор Лапласиана имеет маску размером 3x3. При использование данного оператора для нахождения границ без предварительного сглаживания (применения фильтра Гаусса) в большинстве случаев не удается получить желаемого результата, поэтому приходится предварительно обрабатывать изображение, что ведет к снижению быстродействия алгоритма. В сравнение с операторами, рассмотренными в прошлой лабораторной работе, границы, найденные с помощью оператора Лапласиана не столь явные, но практически отсутствует шумовая составляющая.

При повышении резкости используются изображения с границами, в работе было проведено сравнение выходных изображений при использовании оператора Прюитта и оператора Лапласиана. Значительных отличий при использование различных операторов не выявлено. Из чего можно сделать вывод, что для повышения резкости подойдет изображение с границами, полученное с использованием любого из ранее изученных операторов.

Текст программы

```
#include "stdafx.h"
using namespace std;
using namespace cv;

#define To_Float static_cast<float>

template <typename _Tp_out, typename _Tp>
void Matrix_multiplication(Mat_<_Tp_out> &out, Mat_<_Tp> image_with_frame, int K[9]);
template <typename _Tp>
Mat_<uchar> Frame(Mat_<_Tp> input);
template <typename _Tp>
Mat_<uchar> Convert(Mat_<_Tp> input);
template<typename _Tp>
Mat_<uchar> Formulation(Mat_<_Tp> img, String S, int Kx[9], int Ky[9]);
template <typename _Tp>
void Matrix_addition(Mat_<_Tp> &G, Mat_<_Tp> Gx, Mat_<_Tp> Gy);

int main()
{
	int Laplacian[9] = { -1,-1,-1,-1,8,-1,-1,-1,-1 };
	int Gauss_filter[9] = { 1,2,1,2,4,2,1,2,1 };
	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("LR3");
	String imageName("LR3_2.jpg");
	Mat_<uchar> image_input		// Исходное изображение в оттенках серого
		= imread(imageName.c_str(), IMREAD_GRAYSCALE);
	Mat_<short> image_1		// Изображение для нахождения границ
		= Mat_<short>(image_input.rows,   image_input.cols);
	Mat_<float> image_2		// Изображение для повышения резкости
		= Mat_<float>(image_input.rows,   image_input.cols);
	// Вывод исходного изображения на экран
	imshow("input", image_input);

	/*-------------== Часть 1 ==-------------*/
	// Применение Лапласиана без сглаживания
	Mat_<uchar> image_with_frame	// Изображение с рамкой
		= Frame(image_input);
	Matrix_multiplication(image_1, image_with_frame, Laplacian);
	imshow("Without smoothing", Convert(image_1));
	// Применение фильтра Гаусса 3x3
	// для сглаживания изображения
	Matrix_multiplication(image_1, image_with_frame, Gauss_filter);
	normalize(image_1, image_1,		// Линейное растяжение
		-32767,32767, NORM_MINMAX, -1, Mat());
	for (int i = 0; i < image_1.rows*image_1.cols; i++)
		image_1.at<short>(i) = round(float(image_1.at<short>(i)) / 16);
	imshow("Gauss filter", Convert(image_1));
	// Применение Лапласиана для выделения границ
	image_with_frame		// Изображение после фильтра Гаусса с рамкой
		= Frame(Convert(image_1));
	Matrix_multiplication(image_1, image_with_frame, Laplacian);
	imshow("With smoothing", Convert(image_1));
```

```
	/*------------=== Часть 2 ===-------------*/
	// Повышение резкости
// Применение других операторов для находения границ:
	//Formulation(image_input, "Sobel", Sobel_operator_x, Sobel_operator_y);
	//Formulation(image_input, "Prewitt", Prewitt_operator_x, Prewitt_operator_y);
	//Formulation(image_input, "Roberts", Roberts_operator_x, Roberts_operator_y);
	//image_1 = Formulation(image_input, "Prewitt", Prewitt_operator_x,
Prewitt_operator_y);
	/*------------=== Часть 2 ===-------------*/
	// Повышение резкости
	//image_1 = Formulation(image_input, "Prewitt", Prewitt_operator_x,
Prewitt_operator_y); // раскомментить, если с помощью прюитта
	normalize(image_1, image_1, 10,   // от 1 до 1.3
		13, NORM_MINMAX, -1, Mat());
	for (int i = 0; i < image_2.rows*image_2.cols; i++)
		image_2.at<float>(i) =     // input * Коэф_резкости
		round(To_Float(image_input.at<uchar>(i))*To_Float(image_1.at<short>(i)) /
10);
	imshow("High Boost Filter", Convert(image_2));
	waitKey(0);
	return 0;
}

/**
* Преобразование Mat_<typename> в Mat_<uchar>
*
*	Исходная матрица изображения нормализуется
в формат оттенков пикселей от 0 до 255.
Затем выводится в качестве выходного параметра.
* @input Mat_<typename> исходное изображение
* @return Mat_<uchar>
*/
template <typename _Tp>
Mat_<uchar> Convert(Mat_<_Tp> input)
{
	Mat_<uchar> output(input.rows, input.cols);
	normalize(input, input, 0,        // Линейное растяжение
		255, NORM_MINMAX, -1, Mat());
	output = input;
	return output;
}

/**
* Умножение пикселя на маску
*
*	Исходное изображение с рамкой
умножется на маску K
* @img_wf Mat_<typename> исходное изображение с рамкой
* @K - маска 3x3
* @out Mat_<typename> выходная матрица
*/
template <typename _Tp_out, typename _Tp>
void Matrix_multiplication(Mat_<_Tp_out> &out, Mat_<_Tp> img_wf, int K[9])
{
	for (int i = 1; i < img_wf.rows - 1; i++)
		for (int j = 1; j < img_wf.cols - 1; j++)
			out.at<_Tp_out>(i - 1, j - 1) = round(
				K[0] * To_Float(img_wf.at<_Tp>(i - 1, j - 1)) + K[1] *
To_Float(img_wf.at<_Tp>(i - 1, j)) + K[2] * To_Float(img_wf.at<_Tp>(i - 1, j + 1))
				+ K[3] * To_Float(img_wf.at<_Tp>(i - 1, j - 1)) + K[4] *
To_Float(img_wf.at<_Tp>(i, j)) + K[5] * To_Float(img_wf.at<_Tp>(i - 1, j + 1))
				+ K[6] * To_Float(img_wf.at<_Tp>(i + 1, j + 1)) + K[7] *
To_Float(img_wf.at<_Tp>(i + 1, j)) + K[8] * To_Float(img_wf.at<_Tp>(i + 1, j + 1)));
}
```

```
/**
* Создание изображения с рамкой
*
*	К исходному изображению добавляется по
1 пикселю с каждой стороны. Каждый из которых
заполняется соседним оттенком
* @input Mat_<typename> исходное изображение
* @return Mat_<uchar>
*/
template <typename _Tp>
Mat_<uchar> Frame(Mat_<_Tp> input)
{
	Mat_<_Tp> output(input.rows + 2, input.cols + 2);
	// Копия изображения с белой рамкой:
	for (int i = 1; i <= input.rows; i++)
		for (int j = 1; j <= input.cols; j++)
			output.at<_Tp>(i, j) = input.at<_Tp>(i - 1, j - 1);
	// Заполнение белой рамки соседними цветами:
	// Заполнение вехний и нижний границы
	for (int j = 0; j < output.cols; j++)
	{
		output.at<_Tp>(0, j) = output.at<_Tp>(1, j);
		output.at<_Tp>(output.rows - 1, j) = output.at<_Tp>(output.rows - 2, j);
	}
	// Заполнение левой и правой границы
	for (int i = 0; i < output.rows; i++)
	{
		output.at<_Tp>(i, 0) = output.at<_Tp>(i, 1);
		output.at<_Tp>(i, output.cols - 1) = output.at<_Tp>(i, output.cols - 2);
	}
	return Convert(output);
}

/**
* Нахождение границ в изображение
*
*	С помощью оператора Kx находятся границы
по вертикали. С помощью оператора Ky находятся
границы по горизонтали. Далее с помощью функции
"Matrix_addition" получаем выходное изображение,
содержащее в себе границы исходного изображения.
* @img - Mat_<typename> исходное изображение
* @S - String Название оператора
* @Kx - int[9] оператор X
* @Ky - int[9] оператор Y
* @return Mat_<uchar> изображение с границами
*/
template<typename _Tp>
Mat_<uchar> Formulation(Mat_<_Tp> img, String S, int Kx[9], int Ky[9])
{

	Mat_<_Tp> Gx(img.rows, img.cols);
	Mat_<_Tp> Gy(img.rows, img.cols);
	Mat_<_Tp> G(img.rows, img.cols);
	Matrix_multiplication(Gx, Frame(img), Kx);
	Matrix_multiplication(Gy, Frame(img), Ky);
	Matrix_addition(G, Gx, Gy);
	normalize(Gx, Gx, -32767,		// Линейное растяжение
		32767, NORM_MINMAX, -1, Mat());
	normalize(Gy, Gy, -32767,
		32767, NORM_MINMAX, -1, Mat());
	normalize(G, G, -32767,
```

```
                32767, NORM_MINMAX, -1, Mat());
        // Вывод полученных изображений на экран
        imshow(S + " : Gx", Gx);
        imshow(S + " : Gy", Gy);
        imshow(S + " : G", G);
        return Convert(G);
}

/**
* Сложение матриц
*
*       Сложение матриц, содержащих
границы по оси X и по оси Y
по формуле sqrt(Gx^2+Gy^2)
* @Gx - Mat_<typename> Границы по вертикали
* @Gy - Mat_<typename> Границы по горизонтали
* @G - Mat_<typename> Выходное изображение с границами
*/
template <typename _Tp>
void Matrix_addition(Mat_<_Tp> &G, Mat_<_Tp> Gx, Mat_<_Tp> 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))));
}
```