Санкт-Петербургский политехнический университет Петра Великого
Институт машиностроения, материалов и транспорта
Высшая школа автоматизации и робототехники

ОТЧЁТ

по лабораторной работе №3

Дисциплина: Техническое зрение

Тема: Применение преобразования Фурье для обработки изображений с использованием библиотеки OpenCV

Студент гр. 3331506/70401 *<подпись>* А.А. Ларионов

Преподаватель *<подпись>* В.В. Варлашин

«__»_____2020 г.

Санкт-Петербург
2020

ЗАДАНИЕ

Создать класс *C++*, реализующий прямое и обратное дискретное преобразование Фурье с возможностью вывода спектра. Применить преобразование к полутоновому изображению, сравнить результат и время выполнения с библиотечной функцией.

Реализовать фильтр Баттерворта для низких и высоких частот. В соответствии с фильтром обработать спектр полутонового изображения.

Используя преобразование Фурье, произвести свертку полутонового изображения с ядром следующих фильтров: Собеля (по горизонтали и по вертикали), усредняющего (*Box Filter*) и Лапласа. Сравнить результат с библиотечными функциями соответствующих фильтров.

Используя преобразование Фурье, осуществить поиск по шаблону, т.е. произвести корреляцию полутонового изображения автомобильного номера по очереди с изображениями трех символов и пороговую фильтрацию результирующих изображений.

Для выполнения задания использовать библиотеку *OpenCV 4.0*.

МАТЕМАТИЧЕСКОЕ ОПИСАНИЕ

1. Применение дискретного преобразования Фурье (ДПФ) к изображению как двумерному случаю описывается следующей формулой

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) \cdot e^{-i2\pi(ux/M + vy/N)},$$

где $f(x,y)$ – цифровое изображение размерами $M \times N$; x и y – дискретные переменные пространственной области или координаты на изображении; u и v – дискретные переменные частотной области, причем $u = 0, 1, 2, \dots, M-1$ и $v = 0, 1, 2, \dots, N-1$; u/M и v/N – частоты, на которых берутся отсчеты.

Обратное дискретное преобразование Фурье (обратное ДПФ) позволяет получить $f(x,y)$ при известном $F(u,v)$ по следующей формуле

$$f(x,y) = \frac{1}{MN} \cdot \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) \cdot e^{i2\pi(ux/M + vy/N)},$$

где $x = 0, 1, 2, \dots, M-1$ и $y = 0, 1, 2, \dots, N-1$.

Применяя формулу Эйлера, экспоненту можно записать следующим образом

$$e^{\pm i2\pi(ux/M + vy/N)} = \cos(\alpha) \pm isin(\alpha),$$

где $\alpha = 2\pi(ux/M + vy/N)$.

ДПФ является комплексным и может быть представлено в комплексной форме

$$F(u,v) = Re(u,v) + i \cdot Im(u,v),$$

где $Re(u,v)$ – вещественная часть; $Im(u,v)$ – мнимая часть.

Другая форма записи представляет собой выражение в полярных координатах

$$F(u,v) = |F(u,v)| \cdot e^{i\varphi(u,v)},$$

где $|F(u,v)|$ – амплитуда или спектр; $\varphi(u,v)$ – фазовый угол или фаза.

Спектр и фаза вычисляются по следующим выражениям

$$|F(u,v)| = \sqrt{Re(u,v)^2 + Im(u,v)^2},$$

$$\varphi(u,v) = arctg\left(\frac{Im(u,v)}{Re(u,v)}\right).$$

Однако при вычислении фазы нужно учитывать знаки вещественной и мнимой части для охвата диапазона $[-\pi, \pi]$.

В соответствии с формулой Эйлера из этой формы записи можно перейти к комплексной

$$|F(u,v)| \cdot e^{i\varphi(u,v)} = |F(u,v)| \cdot \cos\big(\varphi(u,v)\big) + i\left(|F(u,v)| \cdot sin\big(\varphi(u,v)\big)\right),$$

$$|F(u,v)| \cdot \cos\big(\varphi(u,v)\big) = Re(u,v),$$

$$|F(u,v)| \cdot sin\big(\varphi(u,v)\big) = Im(u,v).$$

2. Фильтр Баттерворта для низких и высоких частот в дискретных переменных частотной области записывается следующим образом

$$H(u,v)_{low} = \frac{1}{1 + \left(\frac{D(u,v)}{D_0}\right)^{2n}},$$

$$H(u,v)_{high} = \frac{1}{1 + \left(\frac{D_0}{D(u,v)}\right)^{2n}},$$

где $D(u,v)$ – расстояние отсчета от центра координат; D_0 – расстояние среза; n – порядок фильтра.

Обработка спектра одним из вариантов фильтра осуществляется поэлементным умножением

$$|F(u,v)| \cdot H(u,v).$$

3. Фильтр или оператор Собеля использует ядра 3×3

$$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \text{ и } \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}.$$

Фильтр применяют для нахождения приближенных значений производных по вертикали и горизонтали при свертке с изображением

$$G_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} * A,$$

$$G_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} * A,$$

где A – исходное изображение; G_x и G_y – изображения, содержащие приближенные производные по x и y.

Усредняющий фильтр или *Box Filter* имеет следующее ядро

$$\frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

При свертке пикселы результирующего изображения имеют значение, равное среднему значению соседних пикселов исходного изображения

$$G = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} * A.$$

Дискретный оператор Лапласа для двумерного случая задается следующим ядром

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

Оператор Лапласа используется при обработке изображения, например, в задачах выделения границ. Свертка с изображением дает лапассиан, который определяется как сумма вторых производных и вычисляется как сумма перепадов на соседях центрального пиксела

$$G = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} * A.$$

Если представить ядро фильтра как цифровое изображение $f(x,y)_{filter}$ и осуществить ДПФ, то получим $F(u,v)_{filter}$ фильтра. Также поступим и с исходным изображением, получив $F(u,v)_{src}$ из $f(x,y)_{src}$. Свертка исходного изображения и ядра фильтра в пространственной области аналогична умножению в частотной, т.е.

$$f(x,y)_{src} * f(x,y)_{filter} \leftrightarrow F(u,v)_{src} \cdot F(u,v)_{filter}.$$

4. Оценку корреляции (взаимосвязи) двух величин можно произвести посредством взаимной корреляционной функции, которую в случае преобразования Фурье можно записать как

$$\mathcal{F}(f \star g) = \mathcal{F}(f)^* \cdot \mathcal{F}(g),$$

где $\star$ –корреляционная функция; $\mathcal{F}$ – преобразование Фурье; верхний индекс $*$ – комплексное сопряжение; f и g – функции.

Для изображения символа $f(x,y)_{symbol}$ и автомобильного номера $f(x,y)_{src}$ после ДПФ соответственно

$$F(u,v)^*_{symbol} \cdot F(u,v)_{src},$$

где $F(u,v)^*_{symbol} = Re(u,v)_{symbol} - i \cdot Im(u,v)_{symbol}$.

Значение интенсивности пиксела на результирующем изображении зависит от порогового значения и определяется следующей функцией

$$dst(x,y) = \begin{cases} maxValue\ if\ src(x,y) > T(x,y) \\ 0\ otherwise \end{cases},$$

где $src(x,y)$ и $dst(x,y)$ – интенсивность пиксела на исходном и обработанном изображении соответственно, $T(x,y)$ – пороговое значение, $maxValue$ – значение максимальной интенсивности на исходном изображении.

Пороговое значение вычисляется как

$$T(x,y) = maxValue - C,$$

где C – произвольная положительная константа ($C \approx 0{,}01$).

РЕАЛИЗАЦИЯ КЛАССА

Класс *MyFourier* (листинг 1) состоит из заголовочного файла *my_fourier.h* и файла реализации *my_fourier.cpp*.

Свойства класса содержат:

- исходное изображение,
- массивы для прямого и обратного ДПФ исходного изображения,
- массивы для вещественных и мнимых чисел,
- массив для спектра,
- изображение спектра,
- массив для фазы.

Метод класса *setImage* позволяет установить исходное изображение. С помощью методов *getSpectrum* и *getPhase* можно вычислить спектр и фазу. Методы *getImageSpectrum*, *getImageFor* и *getImageRev* возвращают изображение спектра, результат прямого и обратного ДПФ соответственно. Для наглядного отображения спектра используются методы *swapSpectrum* и *logSpectrum*. Причем первый необходим для сдвига данных, в результате которого $F(0,0)$ оказывается в центре частотного прямоугольника, заданного интервалами $[0, M-1]$ и $[0, N-1]$. Прямое и обратное ДПФ обеспечивают методы *transformForward* и *transformReverse*.

Листинг 1 – Класс *MyFourier*

```
class MyFourier
{
public:
    MyFourier();
    ~MyFourier();
private:
    Mat m_src;
    Mat m_dstForward;
    Mat m_dstReverse;

    Mat m_real;
    Mat m_imag;

    Mat m_spectrum;
    Mat m_spectrumImage;
    Mat m_phase;
public:
    int setImage(Mat& image);

    int getSpectrum();
    int getPhase();
    Mat getImageSpectrum();
    Mat getImageFor();
    Mat getImageRev();

    int swapSpectrum();
    int logSpectrum();

    int transformForward();
    int transformReverse();
};
```

СРАВНЕНИЕ С OPENCV

Сравнение работы реализованного ДПФ происходило с библиотечной функцией *dft*, принимающей на вход:

- исходное изображение,
- параметр, указывающий на выполнение прямого или обратного ДПФ.

Сравнение прямого ДПФ проводилось по одному критерию – время выполнения (для класса замерялось время работы метода *transformForward)*.

Сравнение обратного ДПФ проводилось по двум критериям: 1) среднеквадратичная погрешность, высчитываемая по разностному изображению, полученному из результата обратного преобразования и исходного изображения; 2) время выполнения (для класса замерялось время работы метода *transformReverse)*.

Результаты сравнения приведены в таблице 1 и 2.

Таблица 1 – Результаты сравнения прямого ДПФ

Критерии	Функции	
	transformForward	*dft* (прямое)
Время выполнения преобразования (*Debug*)	1492,334 с	0,208 с
Время выполнения преобразования (*Release*)	141,204 с	0,137 с

Таблица 2 – Результаты сравнения обратного ДПФ

Критерии	Функции	
	transformReverse	*dft* (обратное)
Среднеквадратичная погрешность	3,97	0
Время выполнения преобразования (*Debug*)	1416,131 с	0,104 с
Время выполнения преобразования (*Release*)	123,670 с	0,078 с

Результаты обратного ДПФ говорят о верности собственной реализации ДПФ в целом и о его соответствии изложенной выше теории. Отличие среднеквадратичного отклонения от нуля может быть следствием погрешности вычислений. Большое отличие во времени выполнения связано с неоптимальным с точки зрения скорости написанием кода в собственной реализации ДПФ. Способом повышения производительности является использование таблицы для значений экспоненты или алгоритма быстрого преобразования Фурье.

В дальнейшем для работы будут применяться результаты ДПФ, полученные с помощью библиотечной функции.

РЕАЛИЗАЦИЯ ФИЛЬТРА БАТТЕРВОРТА

Реализация фильтра Баттерворта представлена в виде функции *processButterworth* (листинг 2), принимающей на вход:

- исходный массив, содержащий спектр,
- расстояние среза,
- порядок фильтра,
- параметр, указывающий на применение фильтра низких или высоких частот.

Обработка спектра осуществляется в соответствии с изложенной теорией, при этом при неизменной фазе получаются новые значения действительных и мнимых чисел

$$|F(u,v)|' \cdot e^{i\varphi(u,v)} = |F(u,v)|' \cdot \cos\big(\varphi(u,v)\big) + i\left(|F(u,v)|' \cdot sin\big(\varphi(u,v)\big)\right),$$

где $|F(u,v)|'$ – измененный спектр.

Листинг 2 – Функция *processButterworth* и вспомогательная

```
void processButterworth (Mat& specIn, Mat& specOut, int cutOffDist, int order, bool flag)
{
    ...

    if (flag == true) // Low Pass Filter
    {
        for (int u = 0; u < specIn.cols; u++)
        {
            for (int v = 0; v < specIn.rows; v++)
            {
                float d = calculateDist(u, v, specIn.cols, specIn.rows);
                specOut.at<float>(v, u) = specIn.at<float>(v, u) *
              ((float)1 / ((float)1 + (float)pow((d / (float)cutOffDist), (2 * order))));
            }
        }
    }

    if (flag == false) // High Pass Filter
    {
        for (int u = 0; u < specIn.cols; u++)
        {
            for (int v = 0; v < specIn.rows; v++)
            {
                float d = calculateDist(u, v, specIn.cols, specIn.rows);
                specOut.at<float>(v, u) = specIn.at<float>(v, u) *
              ((float)1 / ((float)1 + (float)pow(((float)cutOffDist / d), (2 * order))));
            }
        }
    }

    return;
}

float calculateDist(int x, int y, int cols, int rows)
{
    float dist = 0;

    int cX = cols / 2;
    int cY = rows / 2;

    dist = sqrt((float)pow(abs((x - cX)), 2) + (float)pow(abs((y - cY)), 2));

    return dist;
}
```

СВЕРТКА С ЯДРОМ ФИЛЬТРА

Для получения одинакового числа гармоник и адекватного результата изображение фильтра $f(x,y)_{filter}$ и исходное изображение $f(x,y)_{src}$ предварительно приводятся к одинаковому размеру.

Реализация свертки в частотной области представлена в виде функции *mulImages* (листинг 3), принимающей на вход:

- массив, содержащий ДПФ исходного изображения,
- массив, содержащий ДПФ фильтра,
- параметр, указывающий на выполнение свертки или оценки корреляции.

Вычисления происходят согласно изложенной теории.

Листинг 3 – Функция *mulImages* (свертка)

```
void mulImages(Mat& srcOne, Mat& srcTwo, Mat& dst, bool flag)
{
    ...

    if (flag == false) // Convolution
    {
        for (int u = 0; u < srcOne.cols; u++)
        {
            for (int v = 0; v < srcOne.rows; v++)
            {
                dst.at<Vec2f>(v, u)[0] =
                      srcOne.at<Vec2f>(v, u)[0] * srcTwo.at<Vec2f>(v, u)[0] +
                      (-1) * srcOne.at<Vec2f>(v, u)[1] * srcTwo.at<Vec2f>(v, u)[1];
                dst.at<Vec2f>(v, u)[1] =
                      srcOne.at<Vec2f>(v, u)[0] * srcTwo.at<Vec2f>(v, u)[1] +
                      srcTwo.at<Vec2f>(v, u)[0] * srcOne.at<Vec2f>(v, u)[1];
            }
        }
    }

    ...

    return;
}
```

СРАВНЕНИЕ С OPENCV

Сравнение результата свертки изображения с ядром фильтра в частотной области происходило с библиотечными функциями *Sobel*, *boxFilter* и *Laplacian*.

Функция *Sobel* принимает на вход:

- исходное изображение,
- тип вычисляемой производной (по x или по y).

Размер ядра фильтра по умолчанию равен 3×3.

Функции *boxFilter* и *Laplacian* принимают на вход:

- исходное изображение,
- размер ядра.

Якорная точка усредняющего фильтра по умолчанию находится в центре.

Сравнение проводилось по одному критерию – среднеквадратичная погрешность, высчитываемая по разностному изображению, полученному из результата свертки и библиотечной функции (меньшее значение – лучше).

Результаты сравнения приведены в таблице 3.

Таблица 3 – Результаты сравнения

Критерии	Способы обработки	
	Свертка в частотной области	*Sobel* (по y)
Среднеквадратичная погрешность	37,4	
	Свертка в частотной области	*Sobel* (по *x*)
Среднеквадратичная погрешность	29,2	
	Свертка в частотной области	*boxFilter*
Среднеквадратичная погрешность	5,4	
	Свертка в частотной области	*Laplacian*
Среднеквадратичная погрешность	10,8	

Результаты говорят о некоторой погрешности, т.е. несоответствии свертки изображения с ядром фильтра в частотной области и библиотечной функции. Причиной этому может быть погрешность вычислений.

ПОИСК ПО ШАБЛОНУ

Для получения одинакового числа гармоник и адекватного результата изображение символа $f(x,y)_{symbol}$ и изображение автомобильного номера $f(x,y)_{src}$ предварительно приводятся к одинаковому размеру.

Реализация оценки корреляции в частотной области представлена в виде функции *mulImages* (листинг 4).

Вычисления происходят согласно изложенной теории.

Листинг 4 – Функция *mulImages* (оценка корреляции)

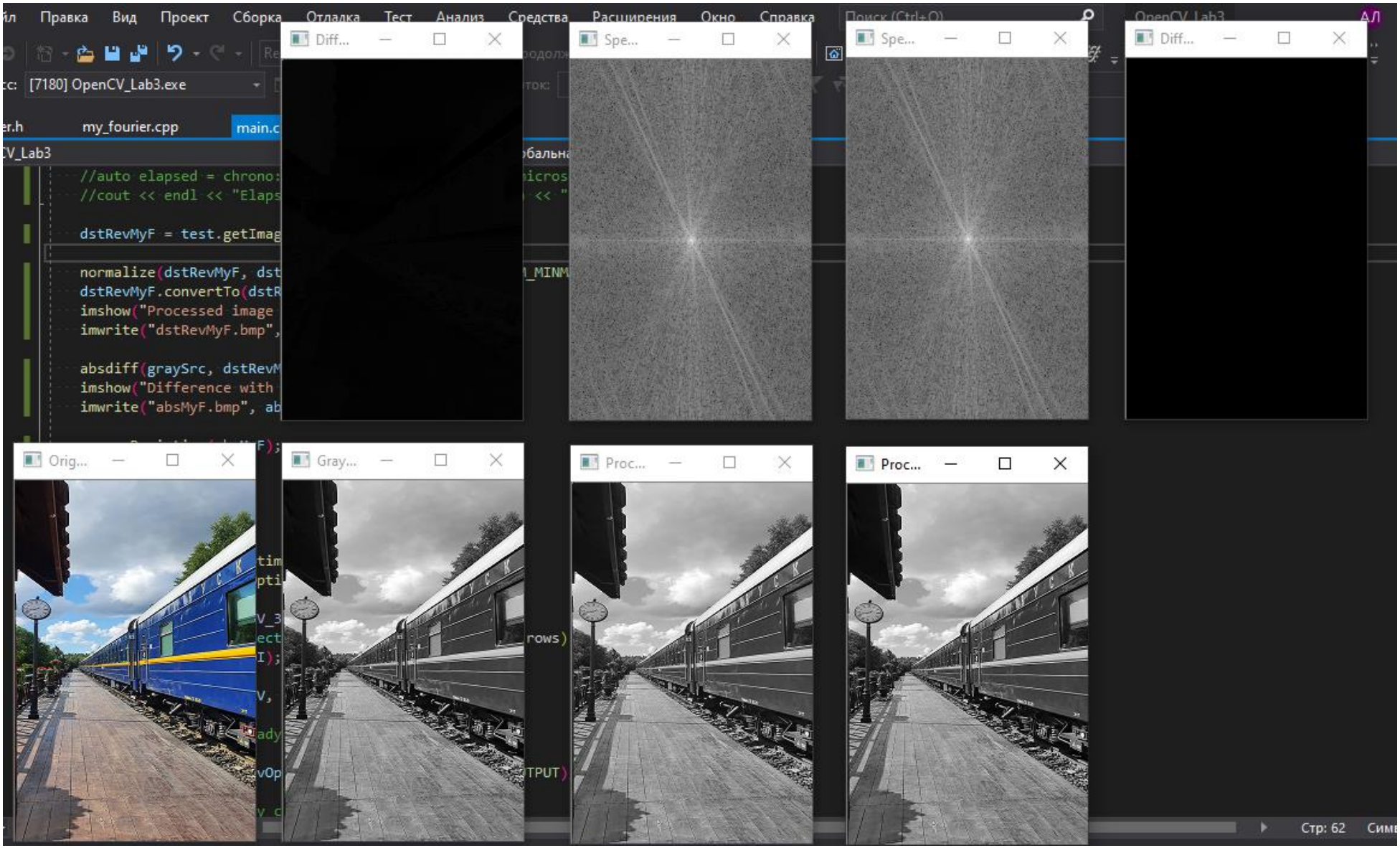
```
void mulImages(Mat& srcOne, Mat& srcTwo, Mat& dst, bool flag)
{
    ...

    if (flag == true) // Correlation/Comparison
    {
        // Conjunction srcTwo
        for (int u = 0; u < srcTwo.cols; u++)
        {
            for (int v = 0; v < srcTwo.rows; v++)
            {
                srcTwo.at<Vec2f>(v, u)[1] *= (-1);
            }
        }

        for (int u = 0; u < srcOne.cols; u++)
        {
            for (int v = 0; v < srcOne.rows; v++)
            {
                dst.at<Vec2f>(v, u)[0] =
                        srcOne.at<Vec2f>(v, u)[0] * srcTwo.at<Vec2f>(v, u)[0] +
                        (-1) * srcOne.at<Vec2f>(v, u)[1] * srcTwo.at<Vec2f>(v, u)[1];
                dst.at<Vec2f>(v, u)[1] =
                        srcOne.at<Vec2f>(v, u)[0] * srcTwo.at<Vec2f>(v, u)[1] +
                        srcTwo.at<Vec2f>(v, u)[0] * srcOne.at<Vec2f>(v, u)[1];
            }
        }

    }

    ...

    return;
}
```

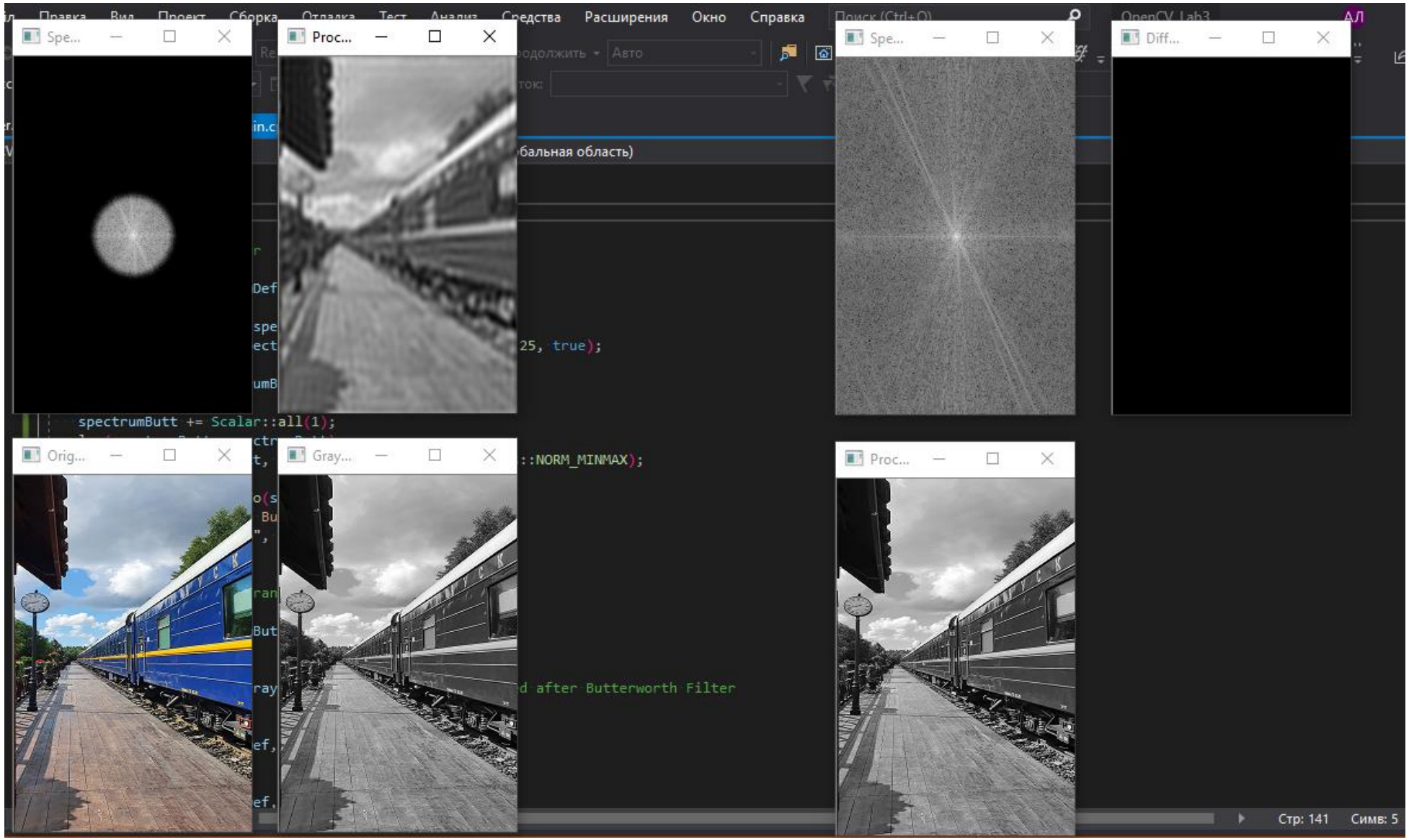
Для наглядного представления результата проводят пороговую фильтрацию результирующего изображения (листинг 5). В качестве

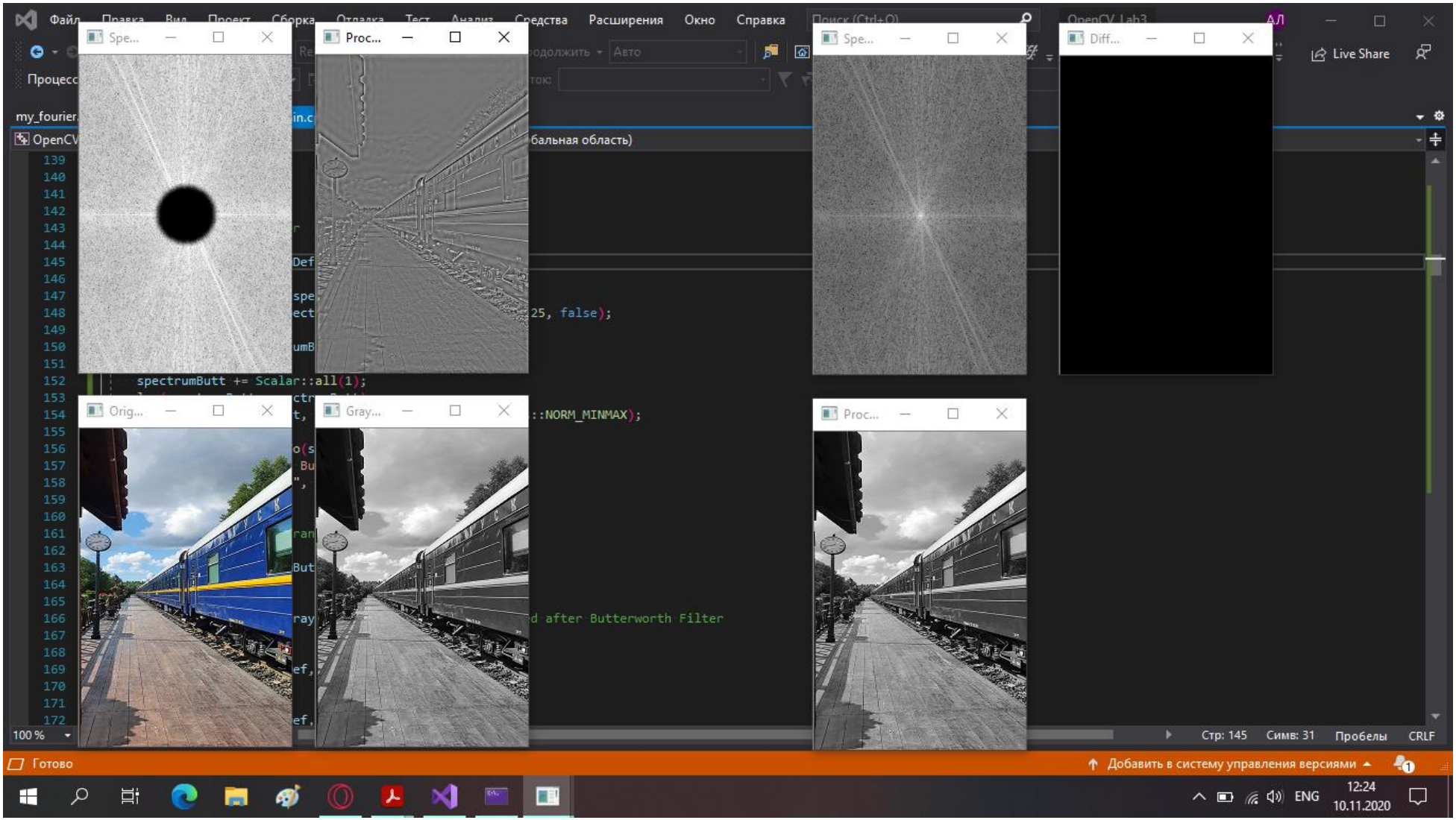
положительной константы C принято значение 0,05. Итогом обработки служат яркие области пикселов на обработанном исходном изображении (монохромном) в местах обнаружения искомого символа.

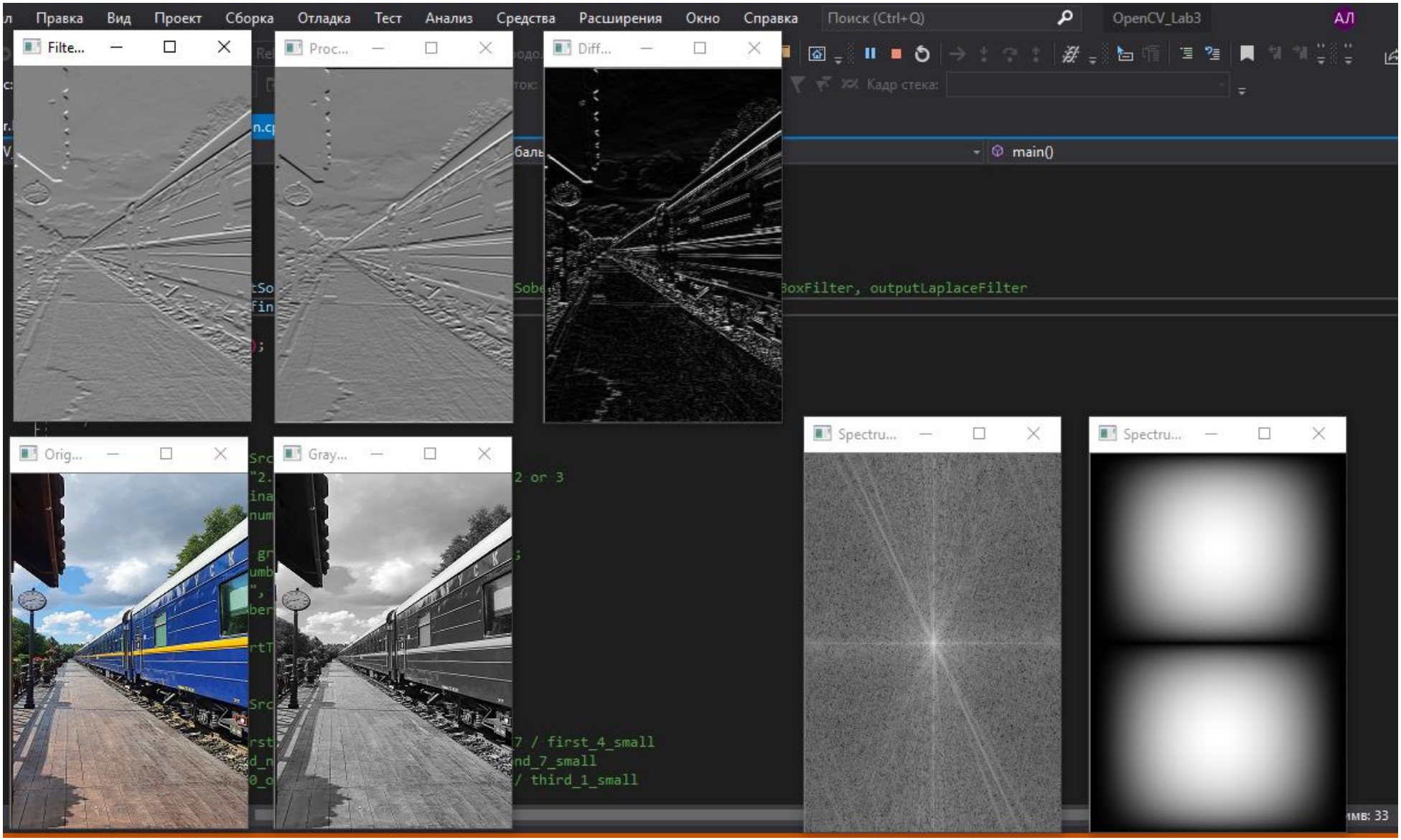
Листинг 5 – Функция *procThreshold*

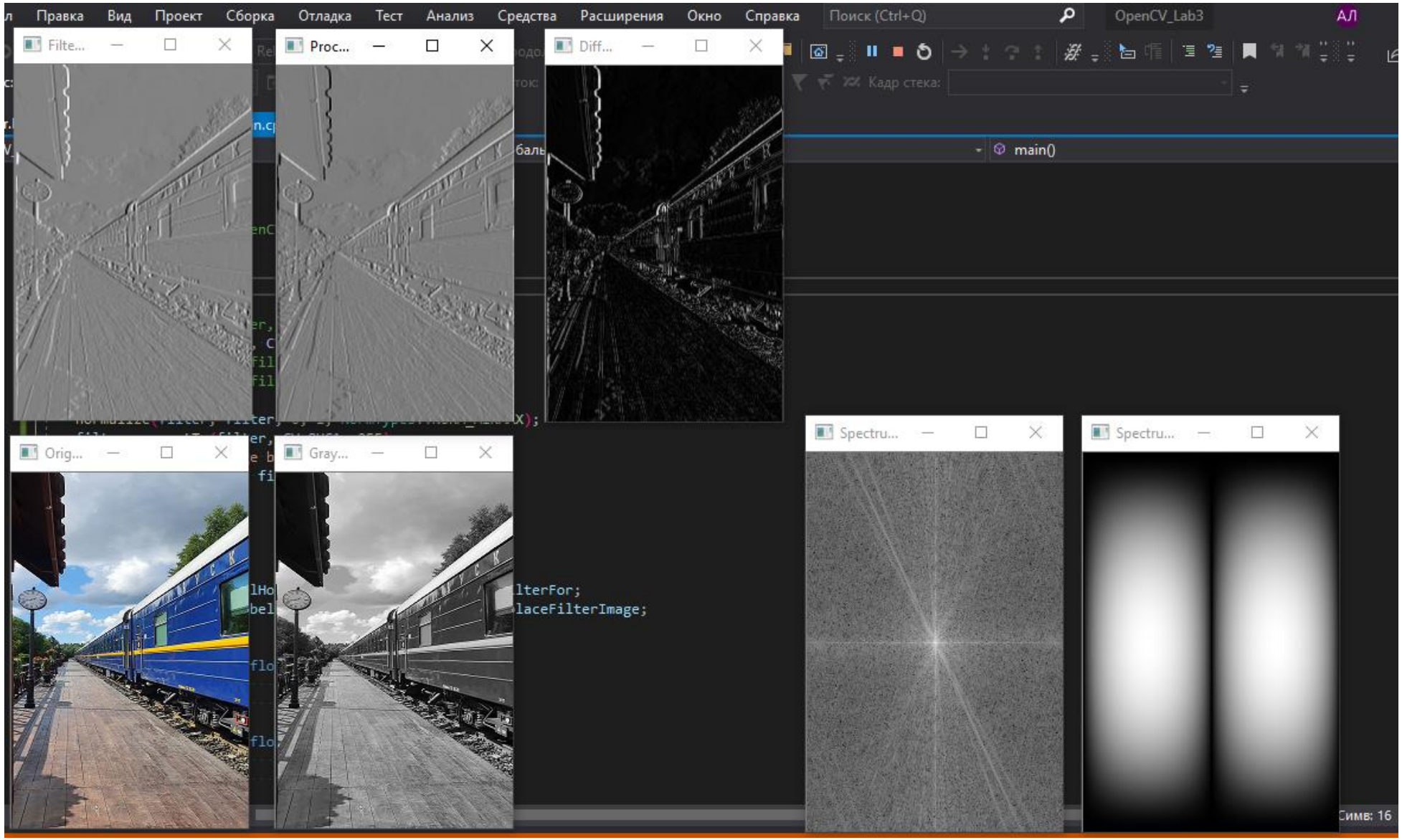
```
void procThreshold(Mat& src, Mat& dst)
{
    ...

    float max = 0;

    for (int u = 0; u < src.cols; u++) // Search max intensity
    {
        for (int v = 0; v < src.rows; v++)
        {
            if (src.at<float>(v, u) > max)
            {
                max = src.at<float>(v, u);
            }
        }
    }

    float T = max - (float)0.05;

    for (int u = 0; u < src.cols; u++) // Threshold
    {
        for (int v = 0; v < src.rows; v++)
        {
            if (src.at<float>(v, u) < T)
            {
                dst.at<float>(v, u) = 0;
            }
            else
            {
                dst.at<float>(v, u) = max;
            }
        }
    }

    return;
}
```

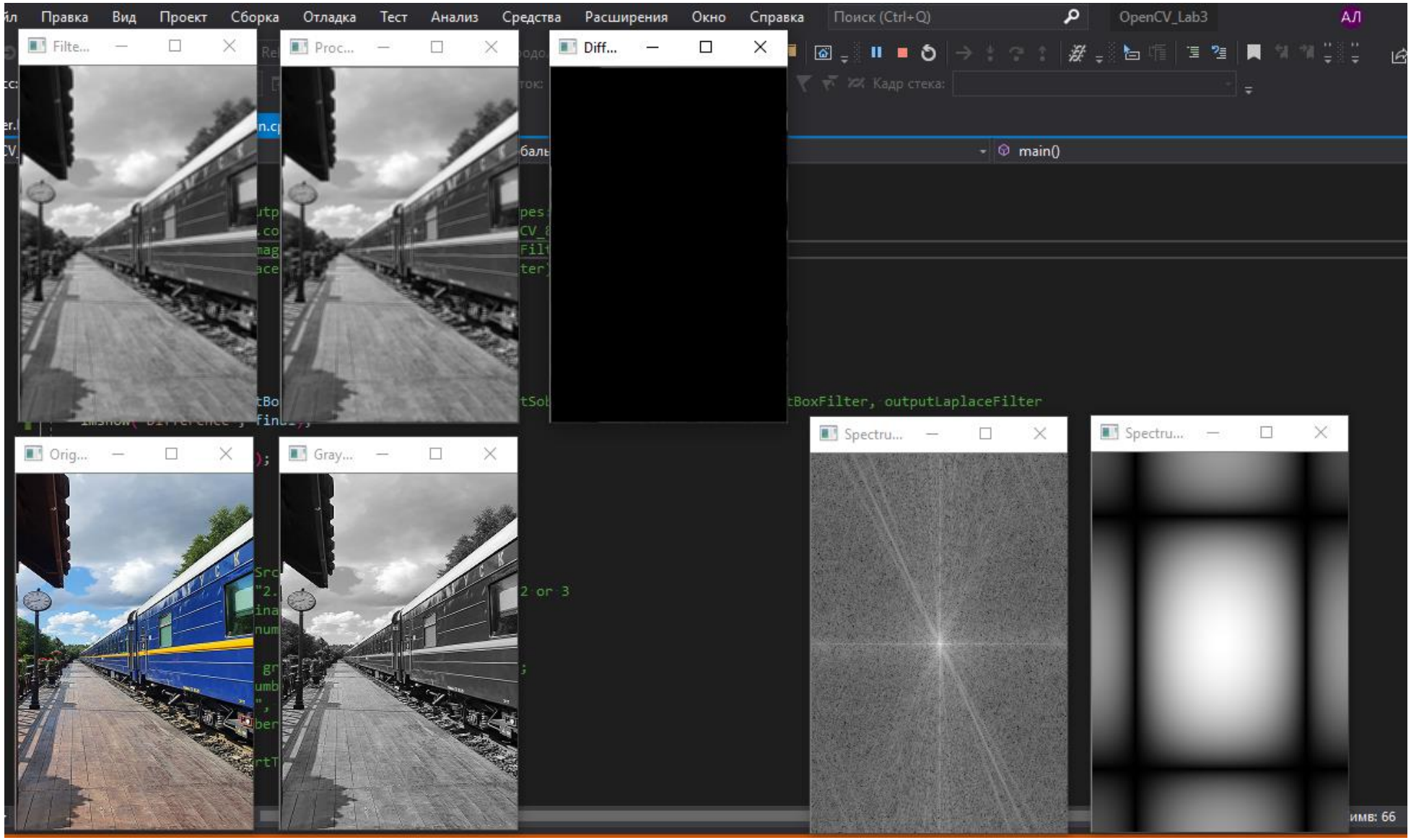

Правка Вид Проект Сборка Отладка Тест Анализ Средства Расширения Окно Справка
[7180] OpenCV_Lab3.exe
my_fourier.cpp
main.c
//auto elapsed = chrono:
//cout << endl << "Elaps
dstRevMyF = test.getImag
normalize(dstRevMyF, dst
dstRevMyF.convertTo(dstR
imshow("Processed image
imwrite("dstRevMyF.bmp",
absdiff(graySrc, dstRevM
imshow("Difference with
imwrite("absMyF.bmp", ab
Diff...
Spe...
Spe...
Diff...
Orig...
Gray...
Proc...
Proc...
Стр: 62

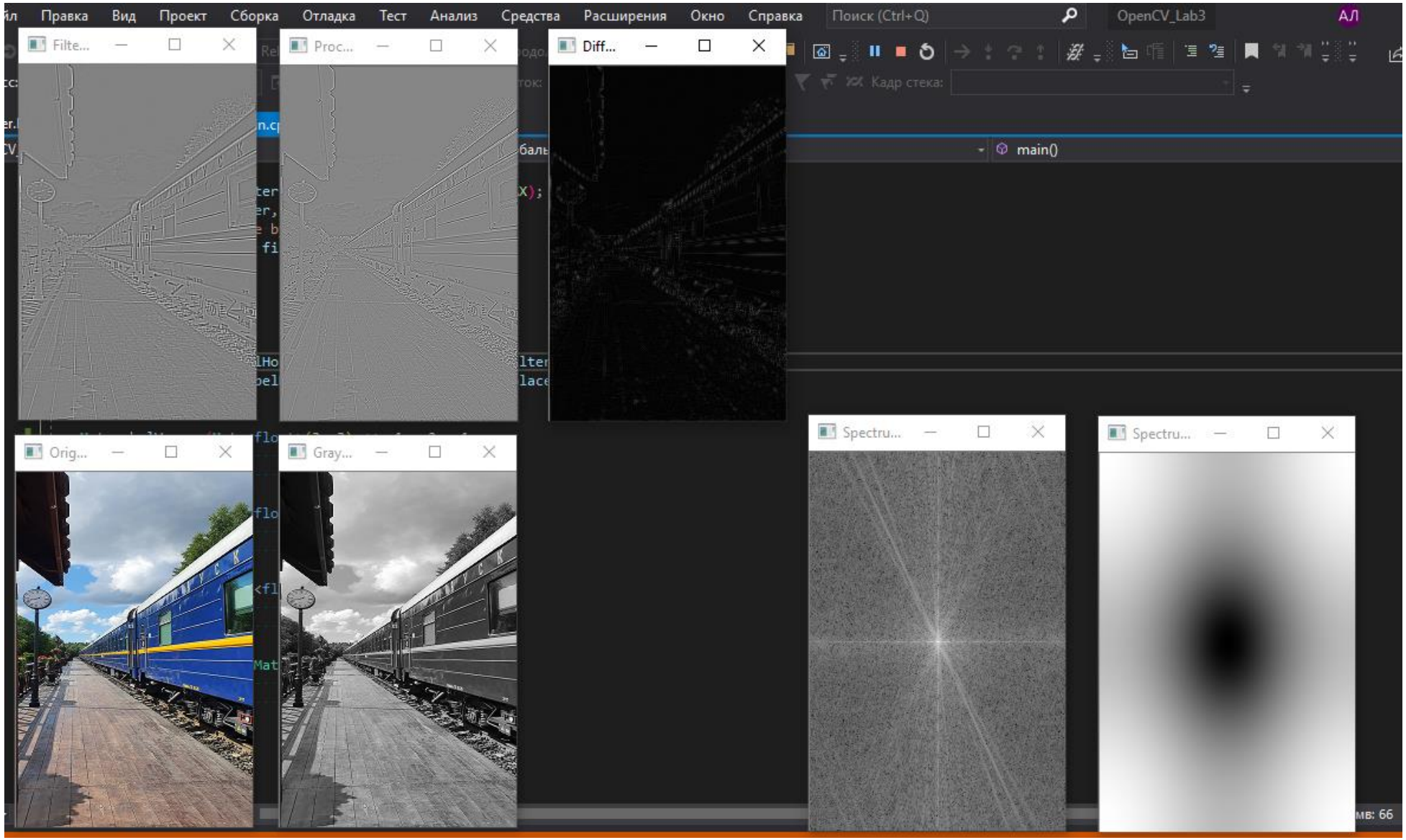

Spe...
Proc...
Spe...
Diff...
Orig...
Gray...
Proc...
Правка Вид Проект Сборка Отладка Тест Анализ Средства Расширения Окно Справка
Поиск (Ctrl+Q)
OpenCV_Lab3
Авто
бальная область)
spectrumButt += Scalar::all(1);
::NORM_MINMAX);
d after Butterworth Filter
Стр: 141
Симв: 5

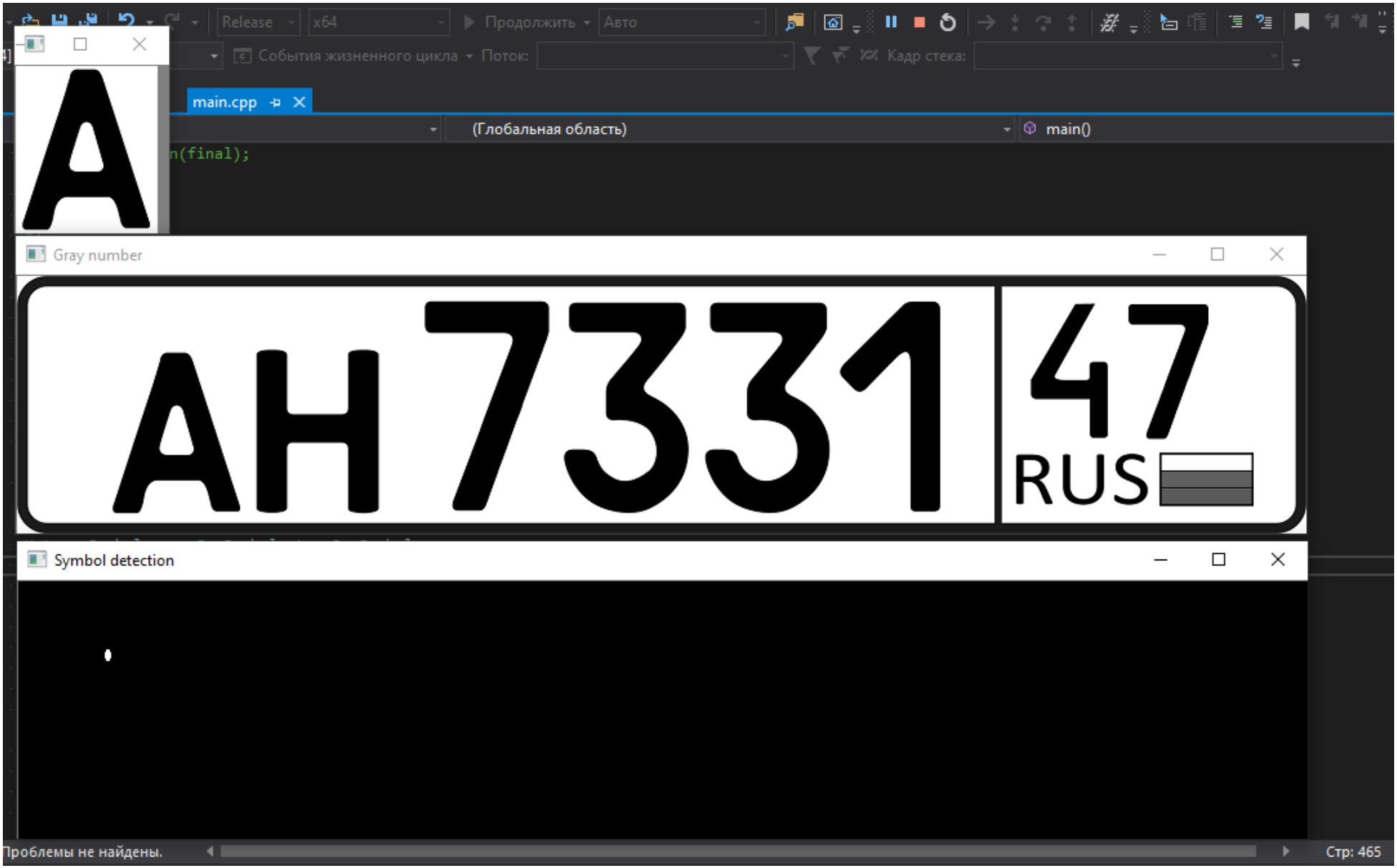

Spe...
Proc...
Spe...
Diff...
Orig...
Gray...
Proc...
Live Share
spectrumButt += Scalar::all(1);
::NORM_MINMAX);
d after Butterworth Filter
Стр: 145
Симв: 31
Пробелы
CRLF
Готово
Добавить в систему управления версиями
ENG
12:24
10.11.2020

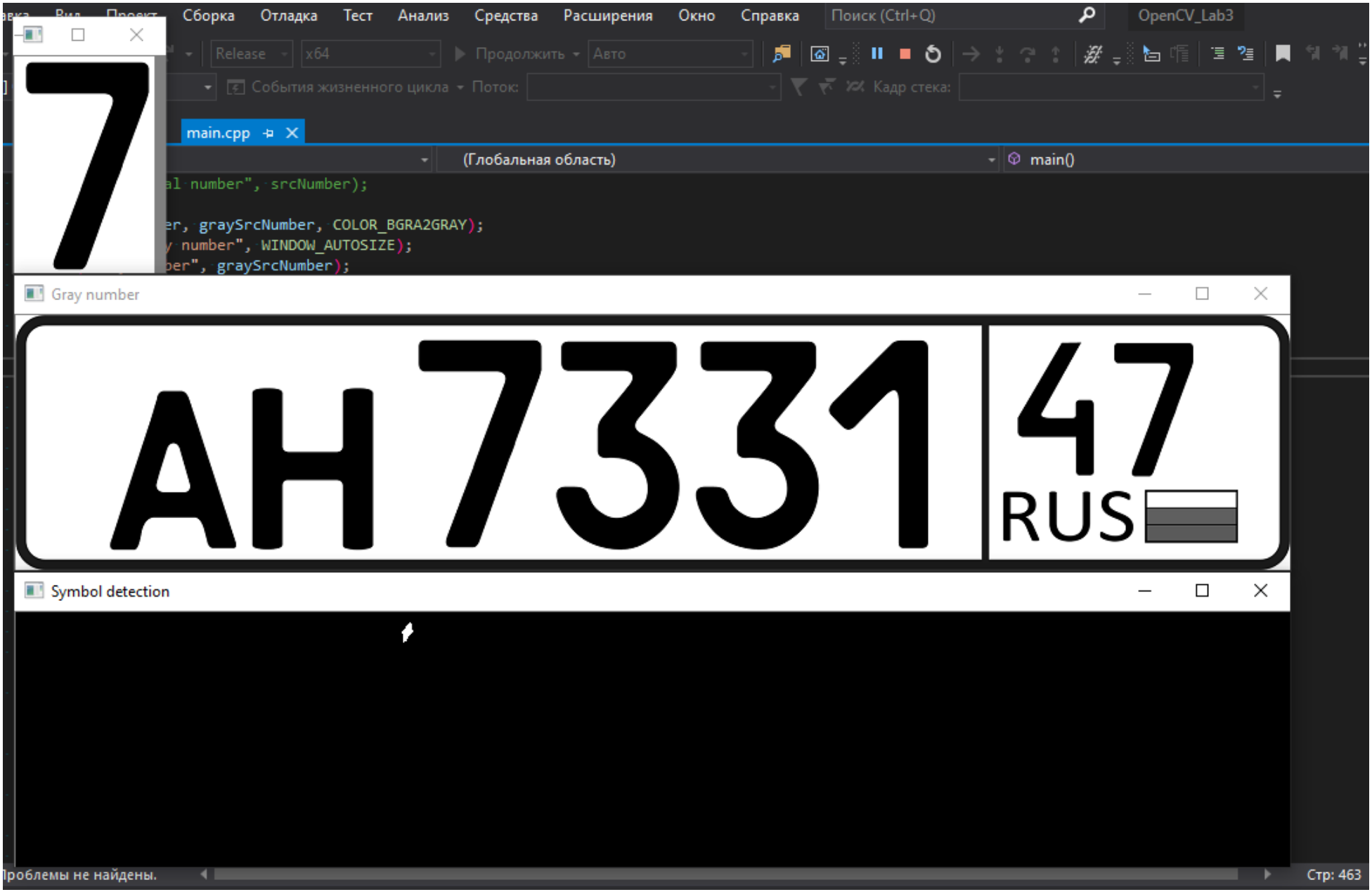

Правка Вид Проект Сборка Отладка Тест Анализ Средства Расширения Окно Справка
Поиск (Ctrl+Q)
OpenCV_Lab3
Filte...
Proc...
Diff...
Orig...
Gray...
Spectru...
Spectru...
main()
BoxFilter, outputLaplaceFilter
2 or 3
7 / first_4_small
nd_7_small
/ third_1_small

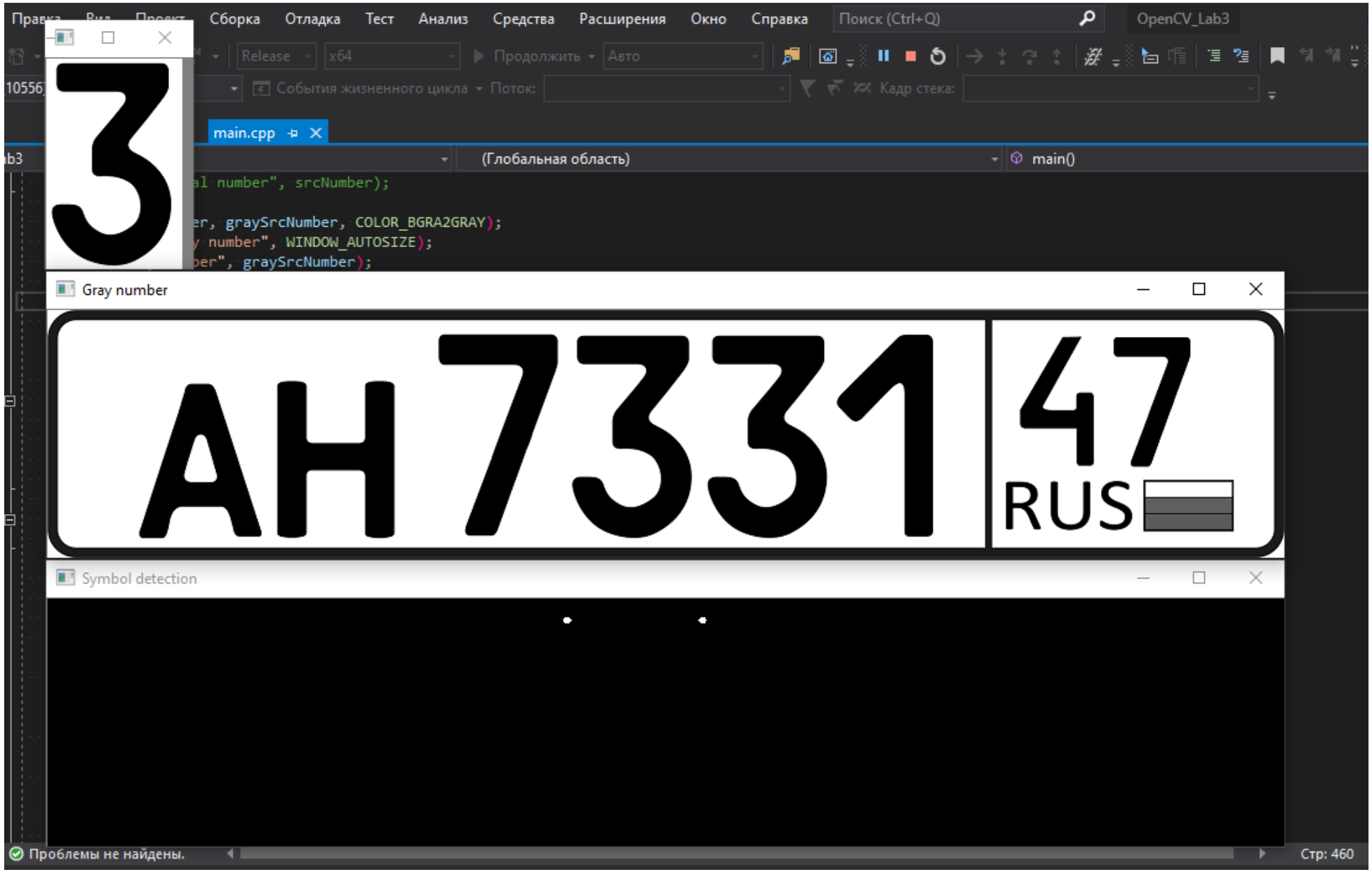

Правка
Вид
Проект
Сборка
Отладка
Тест
Анализ
Средства
Расширения
Окно
Справка
Поиск (Ctrl+Q)
OpenCV_Lab3
Filte...
Proc...
Diff...
Кадр стека:
main()
Orig...
Gray...
Spectru...
Spectru...
lterFor;
laceFilterImage;
Симв: 16


Правка Вид Проект Сборка Отладка Тест Анализ Средства Расширения Окно Справка
Поиск (Ctrl+Q)
OpenCV_Lab3
Filte...
Proc...
Diff...
Orig...
Gray...
Spectru...
Spectru...
main()
Кадр стека:
tBoxFilter, outputLaplaceFilter
2 or 3


Правка
Вид
Проект
Сборка
Отладка
Тест
Анализ
Средства
Расширения
Окно
Справка
Поиск (Ctrl+Q)
OpenCV_Lab3
Filte...
Proc...
Diff...
Orig...
Gray...
Spectru...
Spectru...
main()
Кадр стека:


Release
x64
Продолжить
Авто
События жизненного цикла
Поток:
Кадр стека:
main.cpp
(Глобальная область)
main()
n(final);
A
Gray number
АН 7331
47
RUS
Symbol detection
Проблемы не найдены.
Стр: 465


Сборка
Отладка
Тест
Анализ
Средства
Расширения
Окно
Справка
Поиск (Ctrl+Q)
OpenCV_Lab3
Release
x64
Продолжить
Авто
События жизненного цикла
Поток:
Кадр стека:
main.cpp
(Глобальная область)
main()
al number", srcNumber);
er, graySrcNumber, COLOR_BGRA2GRAY);
y number", WINDOW_AUTOSIZE);
ber", graySrcNumber);
Gray number
АН 7331
47
RUS
Symbol detection
Проблемы не найдены.
Стр: 463


Правка
Сборка
Отладка
Тест
Анализ
Средства
Расширения
Окно
Справка
Поиск (Ctrl+Q)
OpenCV_Lab3
Release
x64
Продолжить
Авто
События жизненного цикла
Поток:
Кадр стека:
main.cpp
(Глобальная область)
main()
al number", srcNumber);
er, graySrcNumber, COLOR_BGRA2GRAY);
y number", WINDOW_AUTOSIZE);
ber", graySrcNumber);
Gray number
АН 7331
47
RUS
Symbol detection
Проблемы не найдены.
Стр: 460