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: Фильтрация изображений в частотной области

Студент гр. 3331506/70401 Самарин А.С.

Преподаватель Варлашин В.В.

« »_____2020 г.

Санкт-Петербург

2020

## Задание

- Реализовать прямое и обратное ДПФ, с возможностью вывода спектра.
- Реализовать фильтр высоких и низких частот в соответствии с заданием.
- Произвести свертку изображения с ядром фильтров: Собеля, усредняющего, Лапласа.
- Провести корреляцию (сравнение) изображений автомобильных номеров по очереди с 3-мя символами.

## Прямое ДПФ

Прямое ДПФ для двумерного массива осуществляется следующим образом:

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

Для реализации создана функция *forwardTransform(Mat inputImage, Mat rezult)*. Данная функция принимает на вход входное одноканальное изображение в формате CV_32FC1 и записывает результат в двухканальное изображение в формате CV_32FC2. В нулевом канале выходного изображения хранится вещественная часть результата, в первом – мнимая.

Для оценки работы функции было найдено время работы и среднеквадратичная погрешность для вещественной и мнимой частей:

| размер изображения | время выполнения, сек | | отклоенние | |
|---|---|---|---|---|
| | библиотечная | моя | Re | Im |
| 150x100 | 0,001 | 10.511 | 0,0449 | 0,0654 |
| 240x340 | 0,001 | 402,213 | 0,0791 | 0,11 |

Для уменьшения ожидания везде вызывается библиотечная функция.

## Вывод спектра

Для нахождения спектра необходимо выполнить следующую операцию:

$$\left|F(u,v)\right|=\left[R^2(u,v)+I^2(u,v)\right]^{1/2}$$

Для вывода спектра используется функция *spectrum(Mat reAndIm, Mat spec)*. Данная функция принимает на вход результат прямого ДПФ и записывает результат в одноканальное изображение CV_32FC1.

Для отображения спектра необходимо результат привести в логарифмический масштаб и преобразовать в CV_8UC1. Для удобного анализа спектра необходимо поменять местами квадранты 1-3 и 2-4.

## Обратное ДПФ

Обратное ДПФ для двумерного массива осуществляется следующим образом:

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

Для реализации создана функция *inverseTransform(Mat reAndIn, Mat rezult)*. Данная функция принимает на вход результат прямого преобразования и записывает результат в одноканальное изображение в формате CV_32FC1.

Для оценки работы функции было найдено время работы и среднеквадратичная погрешность:

| размер изображения | время выполнения, сек | | отклонение |
|---|---|---|---|
| | библиотечная | моя | |
| 150x100 | 0,001 | 7,8 | 0 |
| 240x340 | 0,001 | 271 | 0 |

## Режекторный фильтр Гаусса

Функция фильтра имеет вид:

$$H(u,v) = 1 - e^{-\left[\frac{D^2 - D_0^2}{DW}\right]^2}$$

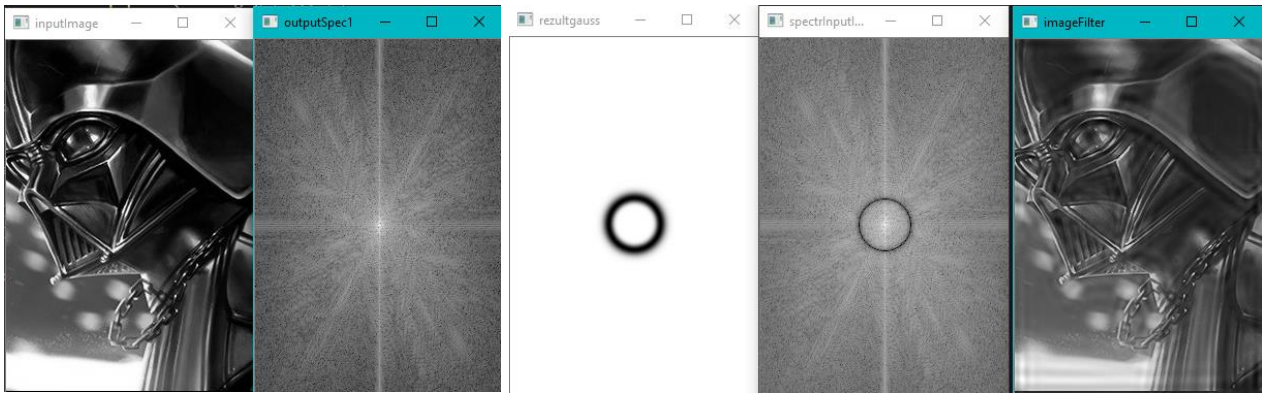
Для применения фильтра используется функция *notchFilter(Mat ReAndIm, Mat newReAndIm, float D0, float W)*. *Mat ReAndIm* – образ входного изображения, *Mat newReAndIm* – переменная для записи образа выходного изображения, , *float D0* – частота среза, *float W* – ширина полосы.

Сперва в функции происходит заполнение фильтра:

```
float x0 = gauss.rows / 2; //центр изображения
float y0 = gauss.cols / 2;

for (int x = 0; x < gauss.rows; x++)
{
	for (int y = 0; y < gauss.cols; y++)
	{
		float D = ((float)x - x0) * ((float)x - x0) + ((float)y - y0)
*((float)y - y0);
		gauss.at<float>(x, y) = 1 - exp(-((D - D0 * D0) / (sqrt(D) * W)) *
((D - D0 * D0) / (sqrt(D) * W)));
	}
}
```

Далее вызывается функция *spectrum* и ее результат перемножается с фильтром.

Входное изображение, спектр, фильтр, спектр после применения фильтра, выходное изображение:



## Свертка с фильтрами

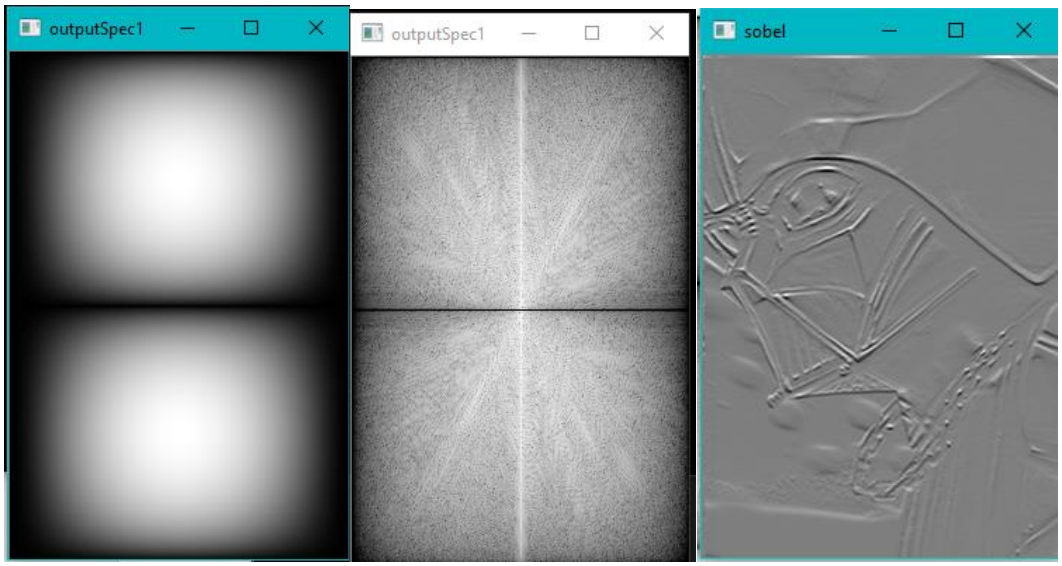
Для произведения свертки изображения с фильтром написаны следующие функции: *laplas(Mat ReAndIm); box(Mat ReAndIm); sobel(Mat ReAndIm, int flag);*

Алгоритм функций:

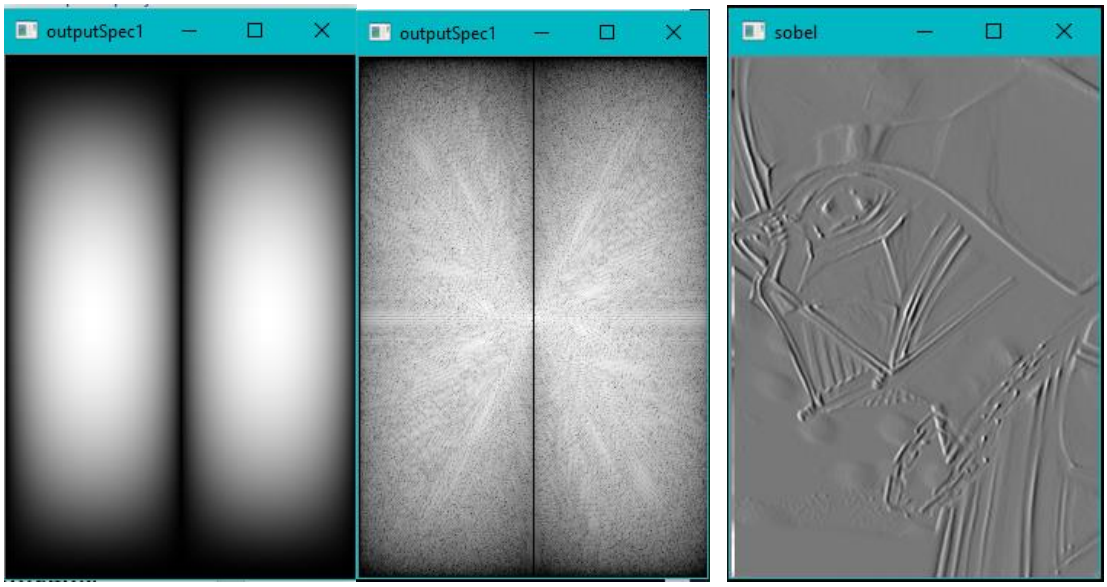
- Создание Mat соответствующего размерам входного изображения и его заполнение коэффициентами фильтра;
- Нахождение образа фильтра;
- Вывод спектра фильтра;
- Свертка входного образа с образом фильтра;
- Вывод спектра результата свертки;
- Обратное преобразование и вывод результата.

*sobel(Mat ReAndIm, int flag)* на вход принимает результат прямого преобразования и флаг. Если flag = 0, то применяется фильтр Собеля по горизонтали, если 1, то по вертикали.

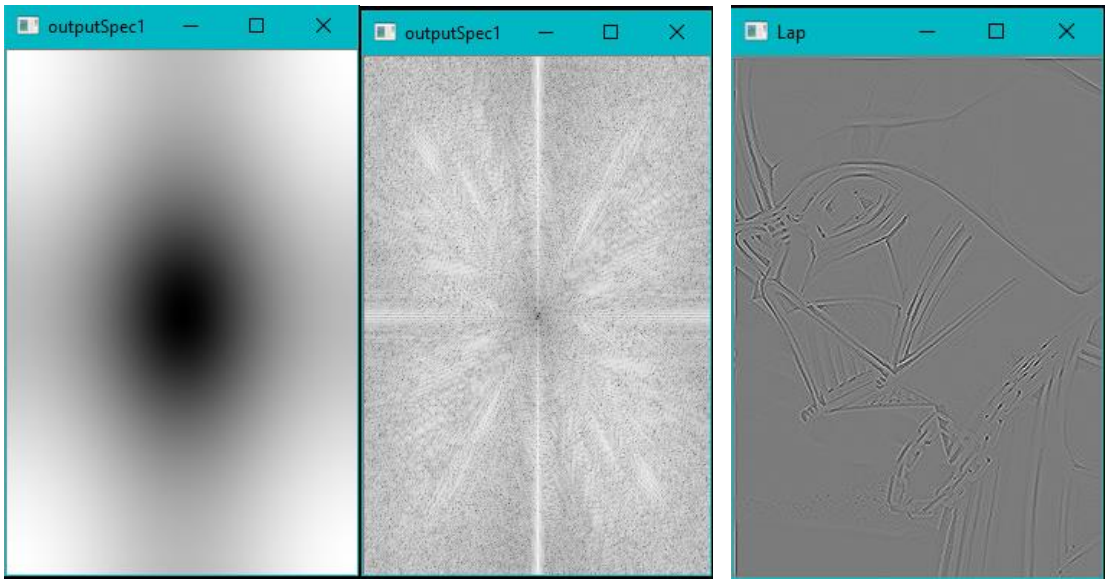
Спектр фильтра Собеля по горизонтали, спектр результата свертки, результат:



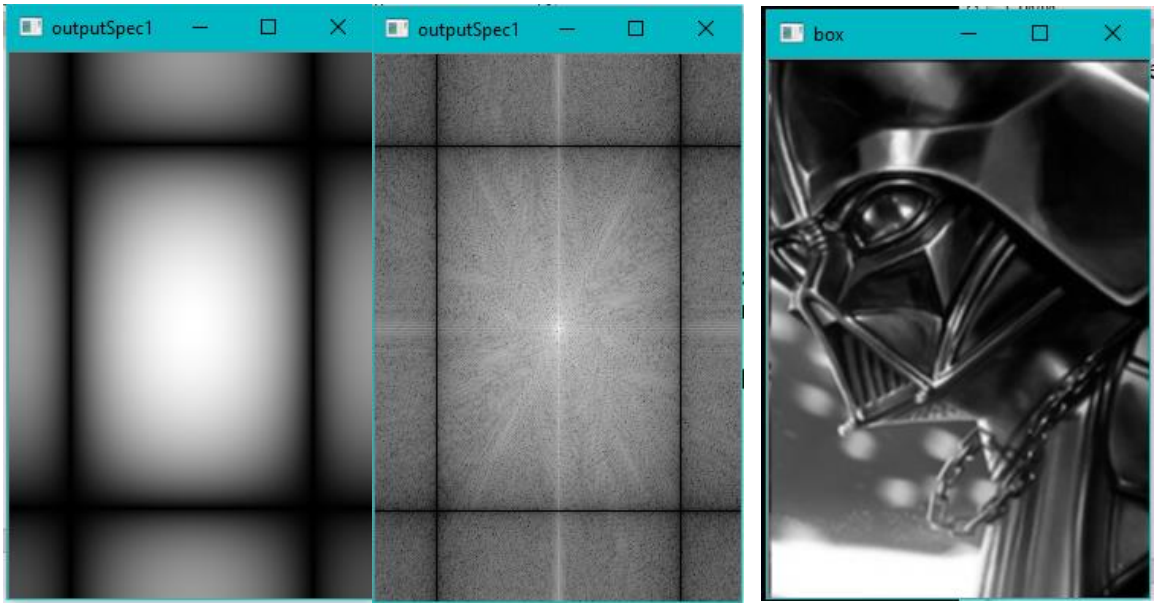
Спектр фильтра Собеля по вертикали, спектр результата свертки, результат:




Спектр фильтра Лапласа, спектр результата свертки, результат:



Спектр прямоугольного фильтра, спектр результата свертки, результат:




## Определение символов на номерах

Для поиска символов на номерах реализована функция num(Mat num, Masymbol). На вход функция принимаем изображение номера и символа. Символ необходимо расширить до размеров номера.

Алгоритм функции:

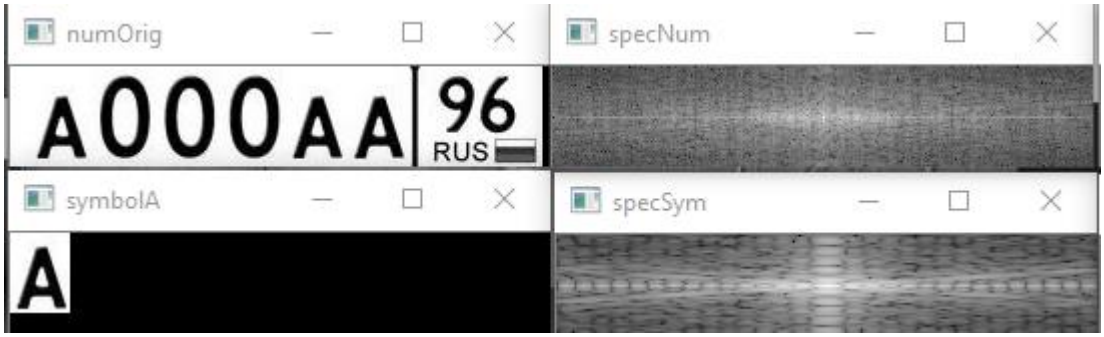
- Нахождение образа номера и символа;
- Вывод спектра номера до корреляции;
- Корреляция образа номера и символа;
- Вывод спектра результата корреляции;
- Обратное преобразование;
- Нормализация и нахождение порогового значения;
- Пороговая фильтрация;
- Вывод итогового изображения.

Корреляция:

```
for (int x = 0; x < num.rows; x++)
{
	for (int y = 0; y < num.cols; y++)
	{
		float a1 = numForward.at<Vec2f>(x, y)[0];
		float b1 = numForward.at<Vec2f>(x, y)[1];
		float a2 = symbolForward.at<Vec2f>(x, y)[0];
		float b2 = -symbolForward.at<Vec2f>(x, y)[1];
		result.at<Vec2f>(x, y)[0] = a1 * a2 - b1 * b2;
		result.at<Vec2f>(x, y)[1] = a1 * b2 + a2 * b1;
	}
```

Поиск символа «А».

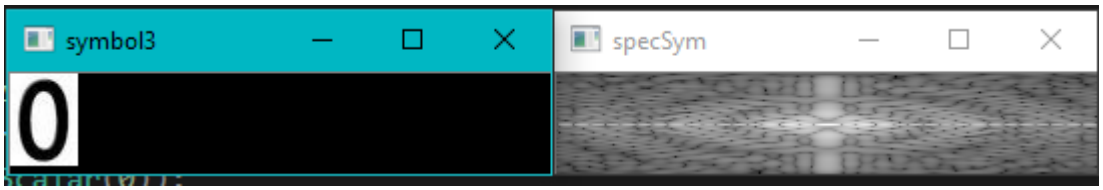
Номер, символ и их спектры до корреляции:




Спектр после корреляции и результат обратного преобразования без фильтрации:


Отклик на символ:

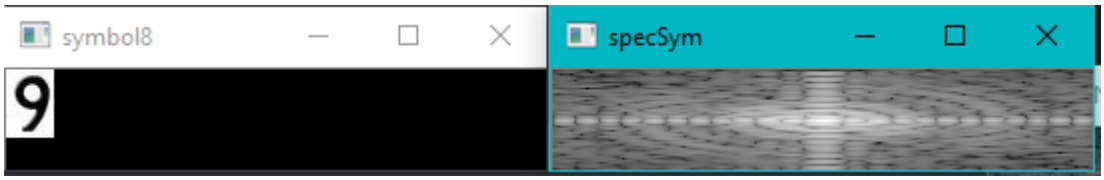

Поиск символа «0».
Символ и его спектр до корреляции:



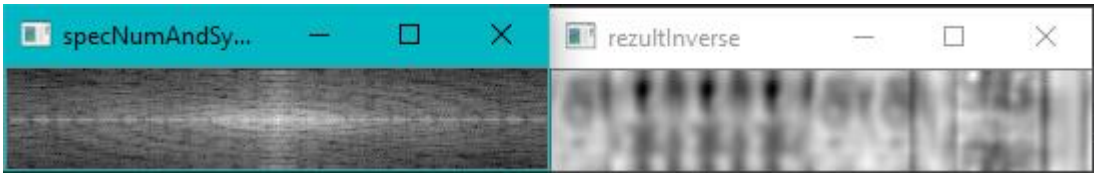

Спектр после корреляции и результат обратного преобразования без фильтрации:


Отклик на символ:

Поиск символа «9».
Символ и его спектр до корреляции:




Спектр после корреляции и результат обратного преобразования без фильтрации:




Отклик на символ:

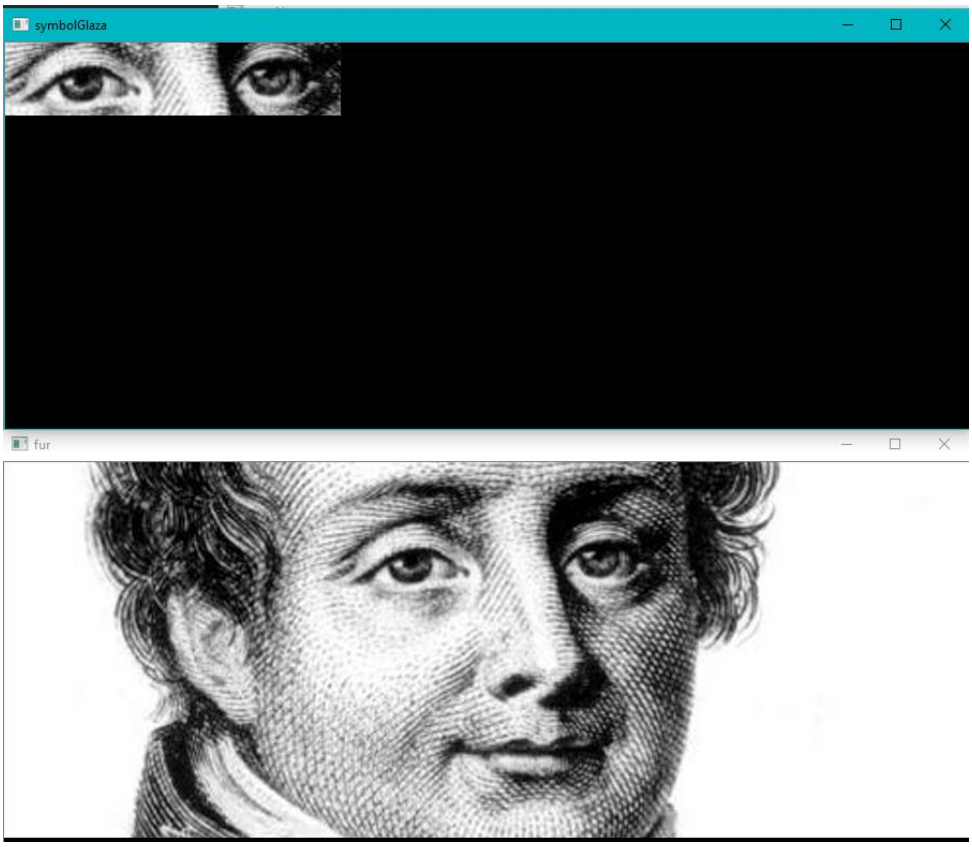
## Защита лабораторной работы

Для нахождения объекта на изображении использовалась операция корреляции. На вход функции подавались образы входных изображений и производилась корреляция:

```
for (int x = 0; x < num.rows; x++)
{
	for (int y = 0; y < num.cols; y++)
	{
		float a1 = numForward.at<Vec2f>(x, y)[0];
		float b1 = numForward.at<Vec2f>(x, y)[1];
		float a2 = symbolForward.at<Vec2f>(x, y)[0];
		float b2 = -symbolForward.at<Vec2f>(x, y)[1];

		result.at<Vec2f>(x, y)[0] = a1 * a2 - b1 * b2;
		result.at<Vec2f>(x, y)[1] = a1 * b2 + a2 * b1;
	}
}
```

Но при таком подходе результат был неточным и отклик мог быть в местах, где его быть не должно. Например:

Входные изображения:



Результат свертки:


После пороговой фильтрации:


Вывод: проводя корреляцию отклик может получиться совершенно не там, где это нужно.

Для решения этой проблемы можно использовать нормированную **кросс-скорреляцию**.

среднее изображение

средний фильтр

$$h[m,n]=\frac{\sum_{k,l}(g[k,l]-\bar{g})(f[m-k,n-l]-\bar{f}_{m,n})}{\left(\sum_{k,l}(g[k,l]-\bar{g})^2\sum_{k,l}(f[m-k,n-l]-\bar{f}_{m,n})^2\right)^{0.5}}$$

Для уменьшения влияния перепадов интенсивности пикселей выполняется вычитание среднего значения. Таким образом меняется диапазон значений. Далее проходит деление на вновь вычисленный диапазон.

Для ускорения работы кросскорреляция проводится в частотной области.

Сперва необходимо найти $\bar{g}$ – среднеарифметическое значение интенсивности в изображении. А далее отнять от каждого пикселя входного изображения эту величину.

Так как исходное изображение находится в формате CV8UC1, то имеет диапазон значений от 0 до 255. И при вычитании $\bar{g}$ может случиться переполнение. Поэтому входные изображения сперва приводятся к типу CV32FC1.

Чтобы найти среднее значение можно воспользоваться функцией void meanStdDev(InputArray src, OutputArray mean, OutputArray stddev, InputArray mask = noArray()).

Получение $g - \bar{g}$ и $f - \bar{f}$:

```
//перевод в float
inputImage1.convertTo(inputImage1, CV_32FC1);
inputImage2.convertTo(inputImage2, CV_32FC1);

//поиск среднего значения
Scalar Mean1;
Mat Std1;
meanStdDev(inputImage1, Mean1, Std1);

Scalar Mean2;
Mat Std2;
meanStdDev(inputImage2, Mean2, Std2);

inputImage1 = inputImage1 - Mean1;
inputImage2 = inputImage2 - Mean2;
```

Далее необходимо расширить фильтр до размеров изображения.

Так как для ускорения работы все происходит в частотной области, то корреляция осуществляется перемножением образа входного изображения на комплексно-сопряженный образ фильтра.

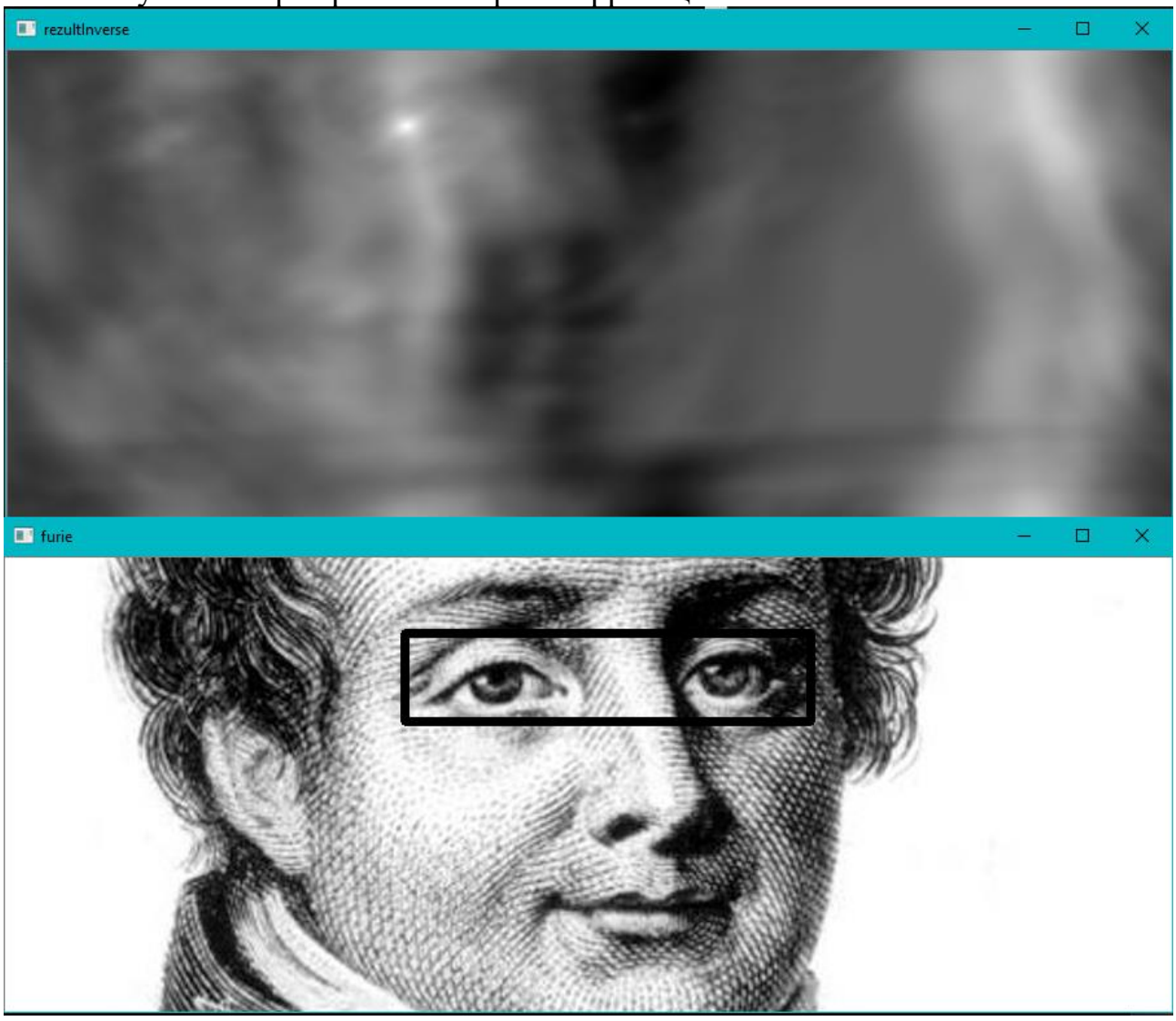
```
//поиск образов
Mat FurForward(inputImage1.size(), CV_32FC2, Scalar(0));
Mat GlazaForward(inputImage1.size(), CV_32FC2, Scalar(0));

cv::dft(inputImage1, FurForward, DFT_COMPLEX_OUTPUT);
cv::dft(Glaza, GlazaForward, DFT_COMPLEX_OUTPUT);

//корреляция
Mat GF;
mulSpectrums(FurForward, GlazaForward, GF, 0, 1);
```

Далее выполняется обратное преобразование и нормализация результата:

```
Mat rezultInverse(GF.size(), CV_32FC1, Scalar(0));
cv::dft(GF, rezultInverse, DFT_INVERSE | DFT_REAL_OUTPUT);
cv::normalize(rezultInverse, rezultInverse, 0, 1, NormTypes::NORM_MINMAX);
```

Так как после прямого преобразования выполняется нормализация результата, то деление результата корреляции можно не проводить.

Результат нормированной кросскорреляции:




Вывод: изменение диапазона корреляции позволяет добиться верного результата, в отличии от простой корреляции. При этом количество операций почти не увеличилось.