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

ОТЧЁТ

по лабораторной работе №2

Дисциплина: Техническое зрение

Тема: Фильтрация изображения 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 адаптивный пороговый фильтр по Гауссу (*Gaussian*) с параметрами: размер ядра 5×5, якорная точка в центре, обработка границ методом отражения. Применить фильтр к полутоновому изображению, сравнить результат и время выполнения с библиотечной функцией. Для выполнения задания использовать библиотеку *OpenCV 4.0*.

МАТЕМАТИЧЕСКОЕ ОПИСАНИЕ

Ключевым элементом фильтра является ядро с нечетным количеством строк и столбцов. В качестве примера и согласно заданию приведено ядро 5×5 с якорной точкой в центре

$$\begin{pmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} \\ C_{21} & C_{22} & C_{23} & C_{24} & C_{25} \\ C_{31} & C_{32} & \boxed{C_{33}} & C_{34} & C_{35} \\ C_{41} & C_{42} & C_{43} & C_{44} & C_{45} \\ C_{51} & C_{52} & C_{53} & C_{54} & C_{55} \end{pmatrix},$$

где C_{ij} – веса, $\boxed{C_{ij}}$ – якорная точка.

Для адаптивного порогового фильтра по Гауссу веса заполняются при помощи двумерной функции Гаусса

$$G(x,y) = \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{(x^2+y^2)}{2\sigma^2}},$$

где σ – среднеквадратичное отклонение.

При этом значение якорной точки равно значению функции $G(0,0)$, а ядро примет следующий вид

$$\begin{pmatrix} G(-2,2) & \cdots & G(2,2) \\ \vdots & \ddots & \vdots \\ G(-2,-2) & \cdots & G(2,-2) \end{pmatrix}.$$

Значение интенсивности пиксела на обработанном 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, $maxValue$ – произвольное значение максимальной интенсивности.

Пороговое значение вычисляется как средневзвешенное исходя из значений интенсивности внутри ядра фильтрации

$$T(x,y) = \frac{1}{\sum_{i,j=1}^{n} C_{ij}} \cdot GaussWeightedSum(x,y) - C,$$

где $GaussWeightedSum(x, y)$ – сумма интенсивностей пикселов исходного изображения внутри ядра фильтрации с соответствующими весами, C – произвольная положительная константа.

Обработка границ осуществляется методом отражения (*border reflect*). В данной работе расширение исходного изображения происходит отражением интенсивностей пикселов от края сначала по горизонтали, а затем по вертикали.

РЕАЛИЗАЦИЯ КЛАССА

Класс *MyFilter* (листинг 1) состоит из заголовочного файла *my_filter.h* и файла реализации *my_filter.cpp*.

Свойства класса определяют следующие параметры:

- исходное и обработанное изображения,
- размер ядра фильтра,
- значения весов ядра C_{ij},
- значение среднеквадратичного отклонения σ,
- значение максимальной интенсивности $maxValue$,
- значение константы C.

При этом размер ядра ограничен нечетными положительными числами с нижней границей равной 3, среднеквадратичное отклонение строго положительно, максимальное значение интенсивности и значение константы лежат в пределах от 0 до 255.

Методы класса *setImage* и *setParam* позволяют установить исходное изображение и все основные параметры для адаптивного порогового фильтра. С помощью метода *getImage* можно вернуть обработанное изображение. Методы *makeKernel* и *processImage* осуществляют вычисление ядра и обработку изображения заданным фильтром соответственно.

Листинг 1 – Класс *MyFilter*

```
class MyFilter
{
public:
    MyFilter();
    ~MyFilter();
private:
    Mat m_src;
    Mat m_dst;

    int m_ksize;
    float m_sigma;
    int m_maxValue;
    int m_constant;

    vector<float> m_coef;
public:
    int setImage(Mat& image);
    int setParam(const int ksize, const float sigma, const int maxValue, const int
constant);

    Mat getImage();

    int makeKernel();
    int processImage();
};
```

СРАВНЕНИЕ С OPENCV

Сравнение работы реализованного адаптивного порогового фильтра происходило с библиотечной функцией *adaptiveThreshold*, принимающей на вход:

- исходное изображение,
- максимальное значение интенсивности,
- вариант метода вычисления порогового значения,
- вариант функции вычисления интенсивности пиксела на обработанном изображении,
- размер ядра,
- значение константы.

Параметры двух сравниваемых вариантов фильтра приведены в таблице 1.

Таблица 1 – Параметры

Параметры	Класс/Функция	
	MyFilter	*adaptiveThreshold*
Размер ядра	5	5
Среднеквадратичное отклонение	2	—
Максимальная интенсивность	255	255
Константа	7	7
Метод вычисления порогового значения	—	*ADAPTIVE_THRESH_GAUSSIAN_C*
Функция вычисления интенсивности пиксела	—	*THRESH_BINARY*

Сравнение проводилось по двум критериям: 1) среднеквадратичное отклонение интенсивности на разностном изображении, полученном из обработанных изображений; 2) время выполнения обработки (для класса замерялось время работы метода *processImage*).

Результаты сравнения приведены в таблице 2.

Таблица 2 – Результаты сравнения

Критерии	Функции	
	processImage	*adaptiveThreshold*
Среднеквадратичное отклонение	≈ 0,092	
Время выполнения обработки (*Debug*)	1804,52 мс	45,22 мс
Время выполнения обработки (*Release*)	63,68мс	2,67 мс

Результаты говорят о некоторой погрешности обработки изображения, причиной которой может быть особенность вычисления среднеквадратичного отклонения σ для функции Гаусса в реализации *OpenCV* или способ обработки границ. Большое отличие во времени выполнения связано с неоптимальным с точки зрения скорости написанием кода в собственной реализации адаптивного порогового фильтра.