МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

**Факультет программной инженерии и компьютерной техники**

**Задание №2**

по дисциплине

“Методы обработки изображений”

Студент:

Гуменник Петр Олегович

Группа P3333

Преподаватель:

Андреев Артем Станиславович

Санкт-Петербург, 2025

### **Цель:**

Изучить базовые функции сглаживания изображений, доступные в библиотеке OpenCV.

Ознакомиться с методами размытия: усредняющее (blur), Гауссово (GaussianBlur), медианное (medianBlur), билатеральное (bilateralFilter).

Настроить собственный проект в Visual Studio.

Заменить изображение по умолчанию (lena.jpg) на своё собственное изображение.

**Процесс выполнения работы:**

#### ****1. Подготовка среды разработки****

* Использована среда **Visual Studio 2022 Community Edition** с компонентами C++.
* Подключена библиотека **OpenCV 4.5.5**.
* Проект настроен вручную:
  + Указаны каталоги include и lib.
  + Добавлены необходимые библиотеки: opencv\_world455.lib, opencv\_world455d.lib.
  + Указан путь к DLL в свойствах отладки.

#### ****2. Анализ и запуск демонстрационного кода OpenCV****

* Изучен пример из официальной документации OpenCV:  
  https://docs.opencv.org/4.x/dc/dd3/tutorial\_gausian\_median\_blur\_bilateral\_filter.html
* В исходном коде используется изображение lena.jpg, загружаемое через samples::findFile().
* Код изучен, запущен и пошагово протестированы все блоки с фильтрами.

#### ****3. Подстановка собственного изображения****

* Удалена строка samples::findFile(...) — она не работает без предустановленных sample-данных.
* Заменено на прямое чтение изображения:

const char\* filename = "D:/Pictures/ilia.jpg";

src = imread(filename, IMREAD\_COLOR);

* Файл был скопирован в проект.
* Убедился, что изображение открывается корректно (проверка через src.empty()).

#### ****4. Изучение фильтров****

В программе протестированы и визуально проанализированы следующие фильтры:

| **Фильтр** | **Описание** |
| --- | --- |
| **Homogeneous Blur** | Простое усреднение — сглаживает резко, убирает шум, но «размывает» детали. |
| **Gaussian Blur** | Сглаживание с весами по нормальному распределению — более естественное размытие. |
| **Median Blur** | Заменяет каждый пиксель на медиану соседей — хорошо работает против «соляного» шума. |
| **Bilateral Filter** | Сохраняет границы, размывая только близкие по цвету пиксели — дорогостоящий, но качественный. |

#include <iostream>

#include "opencv2/imgproc.hpp"

#include "opencv2/imgcodecs.hpp"

#include "opencv2/highgui.hpp"

using namespace std;

using namespace cv;

int DELAY\_CAPTION = 1500;

int DELAY\_BLUR = 100;

int MAX\_KERNEL\_LENGTH = 31;

Mat src; Mat dst;

char window\_name[] = "Smoothing Demo";

int display\_caption(const char\* caption);

int display\_dst(int delay);

int main(int argc, char\*\* argv)

{

namedWindow(window\_name, WINDOW\_AUTOSIZE);

const char\* filename = argc >= 2 ? argv[1] : "ilia.jpg";

src = imread(filename, IMREAD\_COLOR);

if (src.empty())

{

printf(" Error opening image\n");

printf(" Usage:\n %s [image\_name-- default ilia.jpg] \n", argv[0]);

return EXIT\_FAILURE;

}

if (display\_caption("Original Image") != 0)

{

return 0;

}

dst = src.clone();

if (display\_dst(DELAY\_CAPTION) != 0)

{

return 0;

}

if (display\_caption("Homogeneous Blur") != 0)

{

return 0;

}

for (int i = 1; i < MAX\_KERNEL\_LENGTH; i = i + 2)

{

blur(src, dst, Size(i, i), Point(-1, -1));

if (display\_dst(DELAY\_BLUR) != 0)

{

return 0;

}

}

if (display\_caption("Gaussian Blur") != 0)

{

return 0;

}

for (int i = 1; i < MAX\_KERNEL\_LENGTH; i = i + 2)

{

GaussianBlur(src, dst, Size(i, i), 0, 0);

if (display\_dst(DELAY\_BLUR) != 0)

{

return 0;

}

}

if (display\_caption("Median Blur") != 0)

{

return 0;

}

for (int i = 1; i < MAX\_KERNEL\_LENGTH; i = i + 2)

{

medianBlur(src, dst, i);

if (display\_dst(DELAY\_BLUR) != 0)

{

return 0;

}

}

if (display\_caption("Bilateral Blur") != 0)

{

return 0;

}

for (int i = 1; i < MAX\_KERNEL\_LENGTH; i = i + 2)

{

bilateralFilter(src, dst, i, i \* 2, i / 2);

if (display\_dst(DELAY\_BLUR) != 0)

{

return 0;

}

}

display\_caption("Done!");

return 0;

}

int display\_caption(const char\* caption)

{

dst = Mat::zeros(src.size(), src.type());

putText(dst, caption,

Point(src.cols / 4, src.rows / 2),

FONT\_HERSHEY\_COMPLEX, 1, Scalar(255, 255, 255));

return display\_dst(DELAY\_CAPTION);

}

int display\_dst(int delay)

{

imshow(window\_name, dst);

int c = waitKey(delay);

if (c >= 0) { return -1; }

return 0;

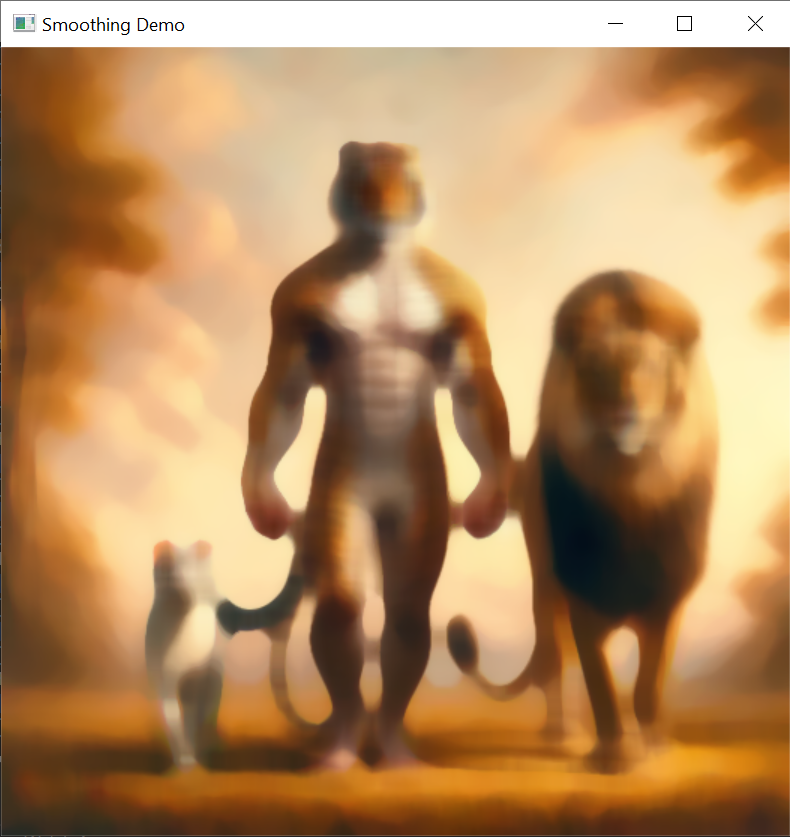
}

**4. Результаты**

Программа успешно скомпилирована и запущена.

Отображены этапы: исходное изображение → последовательное применение всех фильтров.

Продемонстрировано влияние изменения размера ядра фильтра от 1 до 31 (с шагом 2).

Результаты визуально проанализированы: каждый тип фильтра даёт уникальный эффект.

**5. Выводы**

В ходе работы:

Получены практические навыки работы с функциями фильтрации изображений в OpenCV.

Изучена работа популярных сглаживающих фильтров и их поведение при разных параметрах.

Осуществлена замена стандартного изображения на собственное, без использования аргументов командной строки.

Настроен собственный проект в Visual Studio, что пригодится для будущих проектов компьютерного зрения.