МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ

Звіт

З лабораторної роботи №9

На тему «Обробка зображення»

З дисципліни «Основи проектування систем штучного інтелекту та розпізнавання образів»

Підготував: Коваль Ігор Леонідович

Студент навчально-наукового інституту автоматики, кібернетики та обчислювальної техніки

група КН-51м

Рівне 2022

Мета роботи: Навчитися виконувати геометричне вирівнювання; морфологічні операції ерозії, дилатації, відкриття, замикання та застосовувати фільтри згладжування.

Теоретичні відомості

Геометричне вирівнювання

Іноді постає проблема вирівнювати зображення, наприклад коли камера дивиться збоку на площину. Тоді слід вирівняти отриману картинку так, ніби камера дивиться строго зверху.

Виділяються декілька основних видів перетворення: перспективне перетворення та білінійне.

Фільтри згладжування

Гаусівський фільтр дає саме природнє зображення. Медіанний фільтр виділяє області одного кольору, видаляє маленькі деталі. Фільтр усереднення підкреслює маленькі деталі.

Результат роботи програми

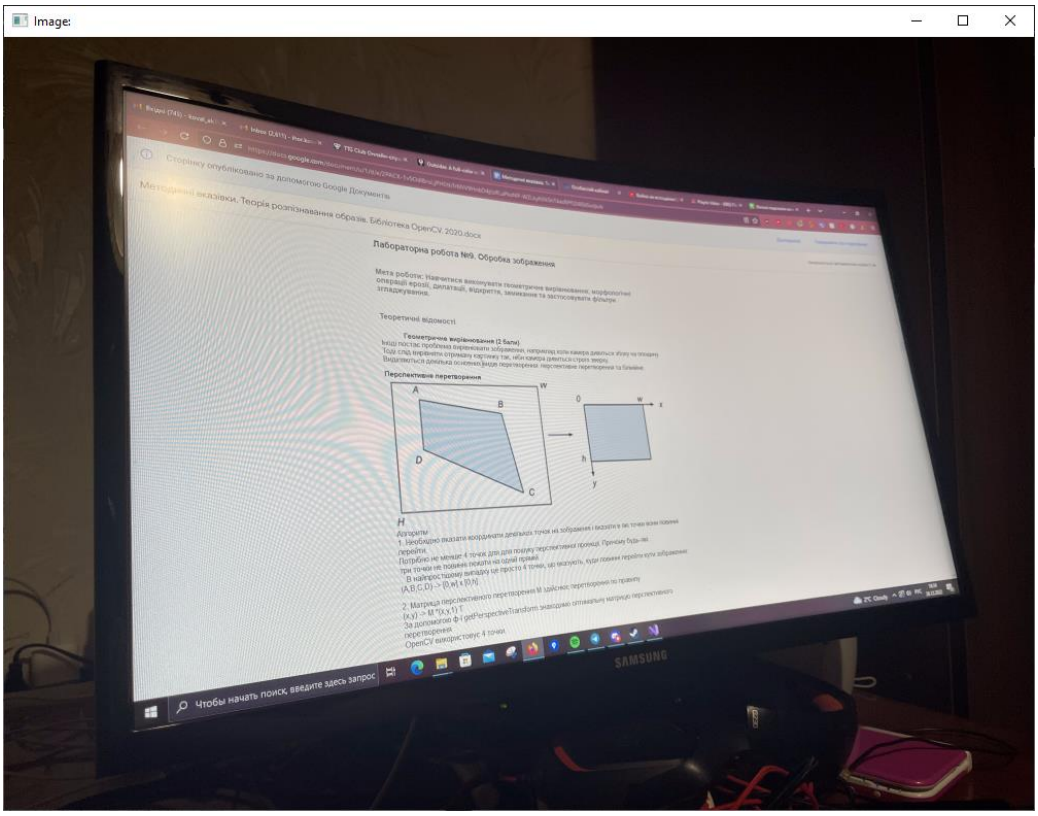

Рис. 1. Вихідне зображення

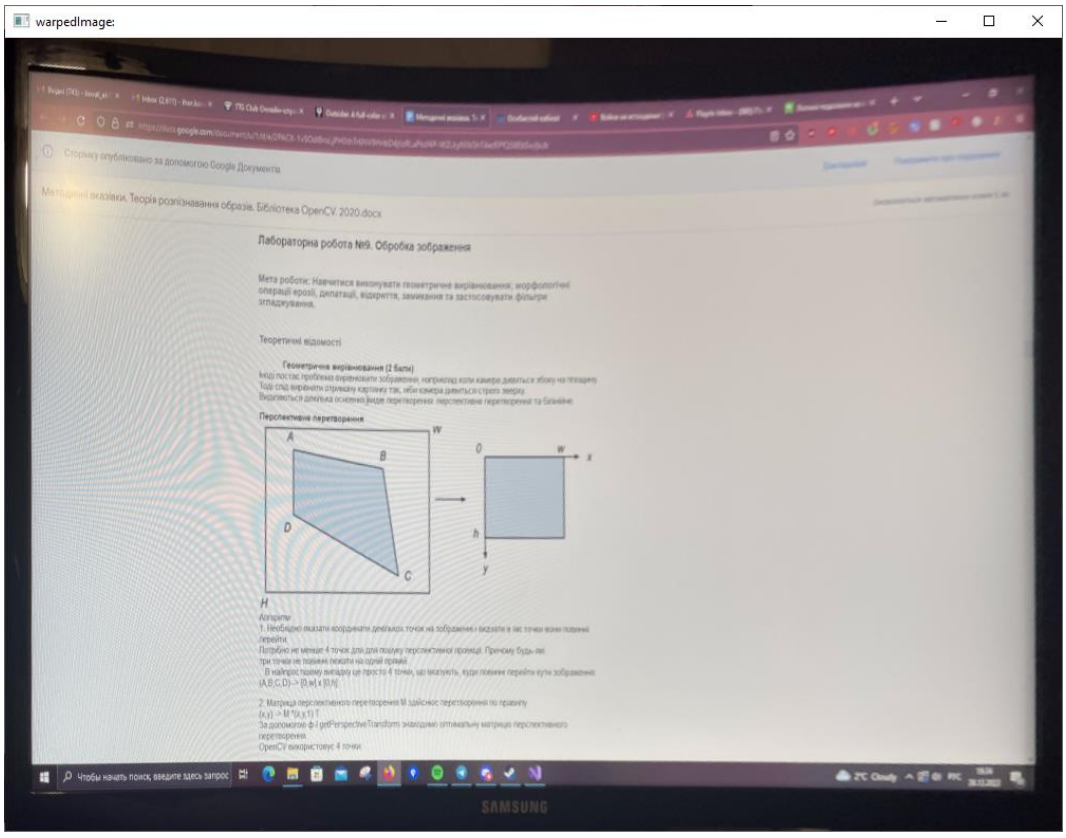

Рис. 2. Деформоване зображення

Видно, що права частина трохи розмита, адже в оригінальному зображенні для неї було менше інформації(пікселів).

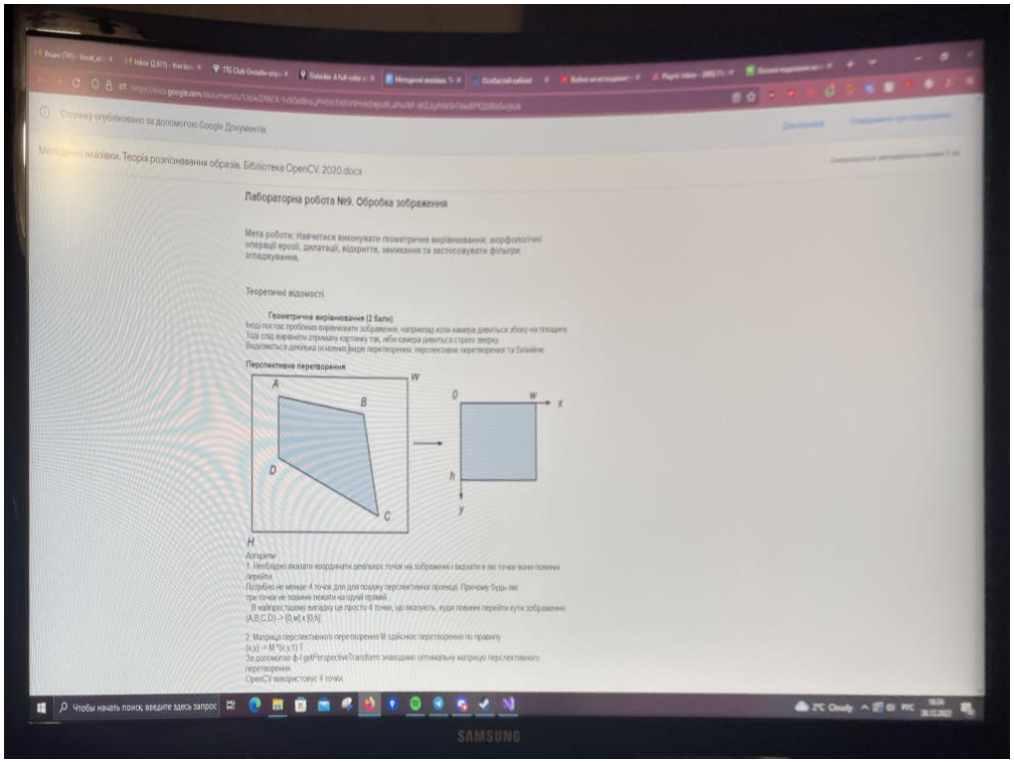

Рис. 3. Кадр №1 з процесу Гаусівського розмиття.

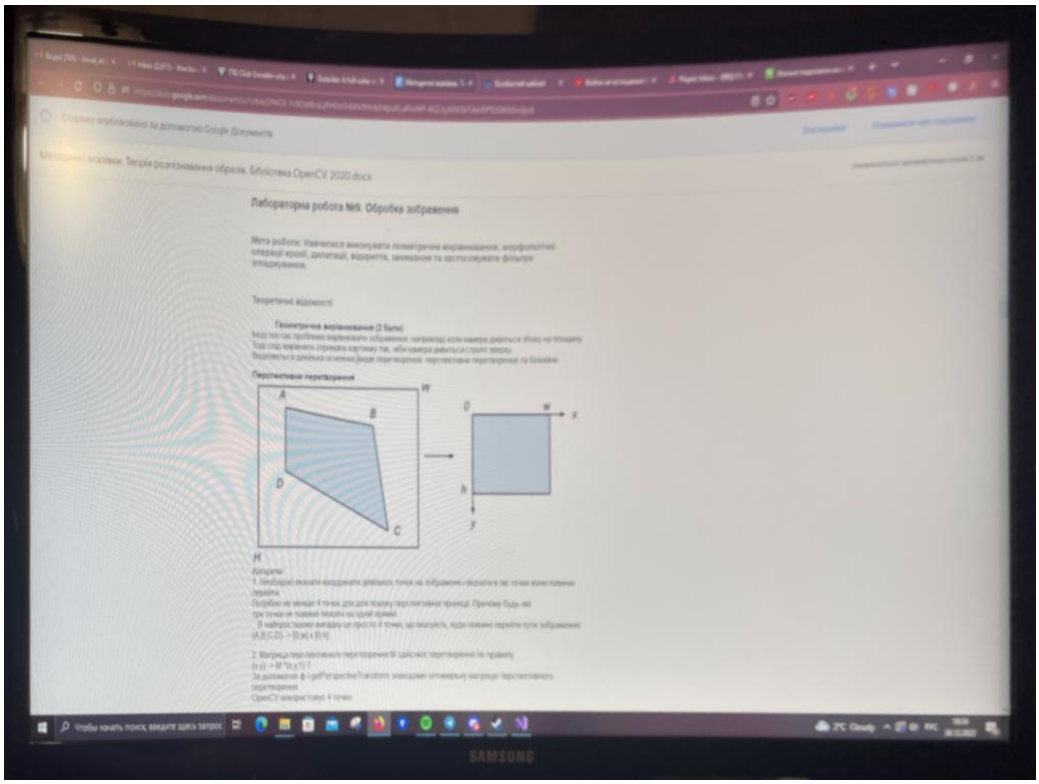

Рис. 4. Кадр №25 з процесу Гаусівського розмиття.

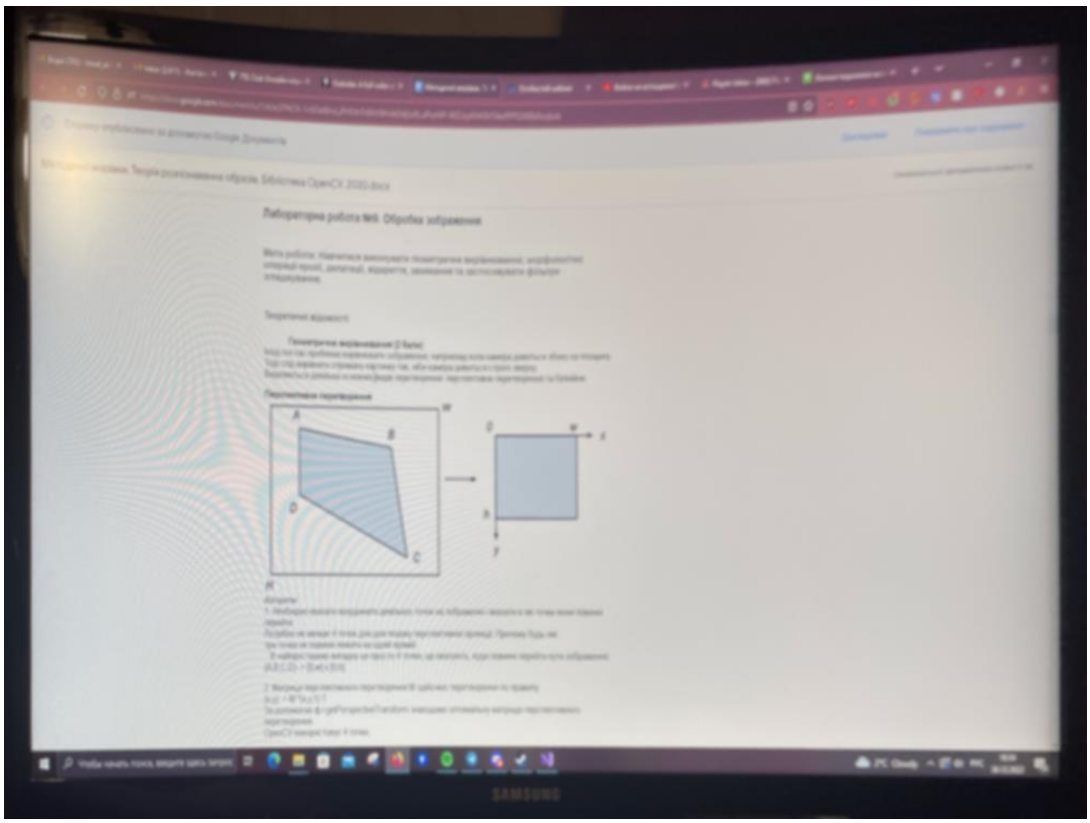

Рис. 5. Кадр №50 з процесу Гаусівського розмиття.

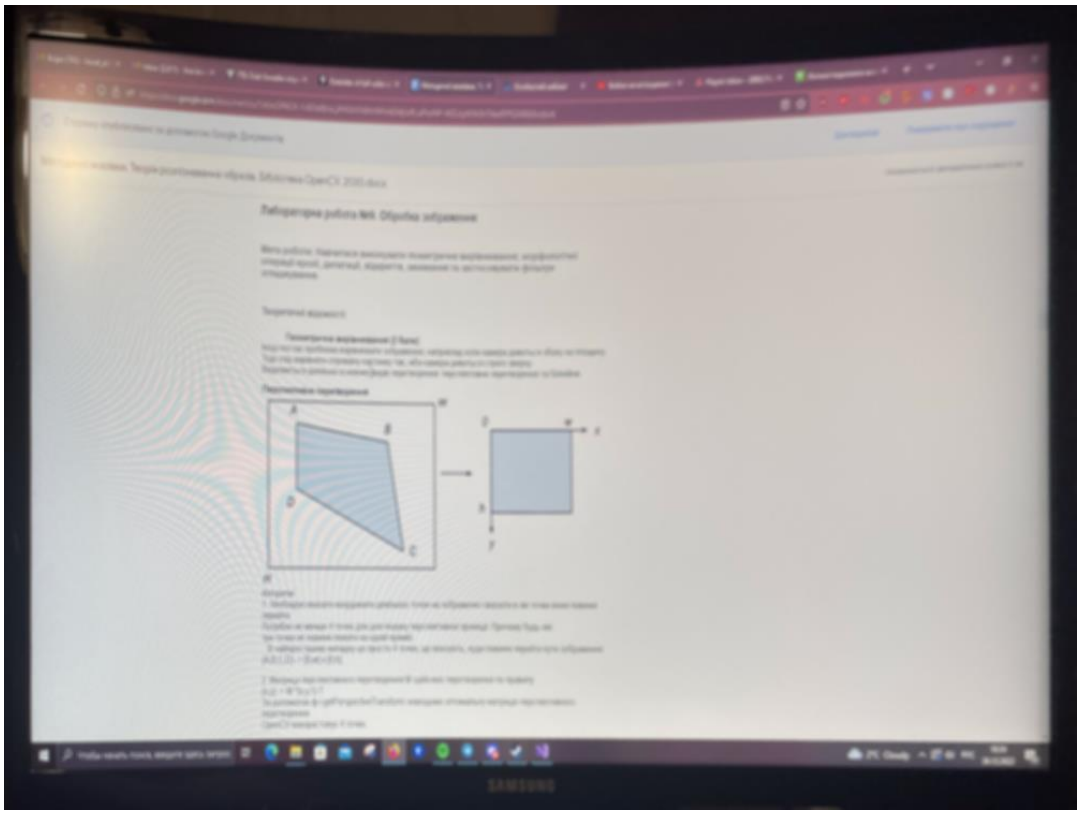

Рис. 6. Кадр №75 з процесу Гаусівського розмиття.

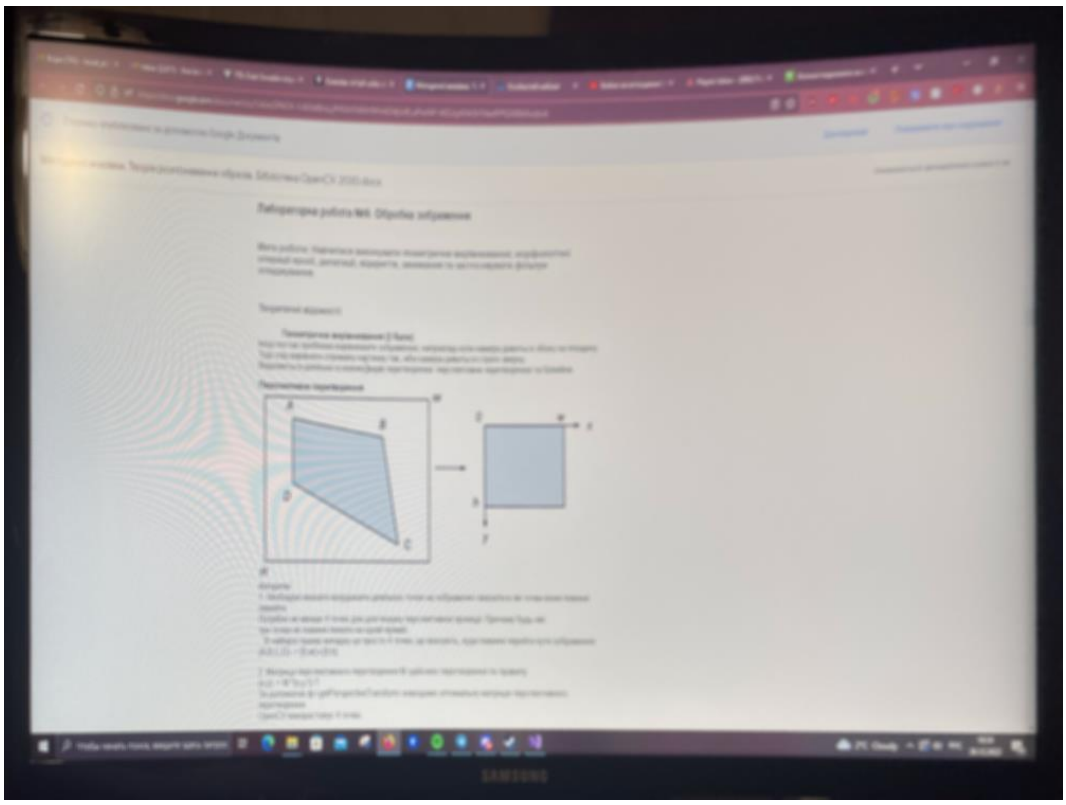

Рис. 7. Кадр №100 з процесу Гаусівського розмиття.

Відео плавного розмиття можна знайти за посиланням https://youtu.be/UddALoUHujQ

Код програми

```
#include <opencv.hpp>
#include <core.hpp>
#include <highgui.hpp>
#include <algorithm>

cv::Mat showWarpedImage(cv::Mat originalImage);
void showGaussImage(cv::Mat originalImage);

int main()
{
    cv::Mat image = cv::imread("1.png");

    cv::namedWindow("Image:", 1);
    cv::imshow("Image:", image);

    auto warped = showWarpedImage(image);

    showGaussImage(warped);

    cv::waitKey();

    return 0;
}

cv::Mat showWarpedImage(cv::Mat originalImage) {
    float h = static_cast<float>(originalImage.size().height);
    float w = static_cast<float>(originalImage.size().width);

    float xDiff = 0.f;
    const int K = 4;
    cv::Point2f src[K];
    cv::Point2f dst[K];
    src[0] = cv::Point2f(88.f, 23.f);
    src[1] = cv::Point2f(854.f, 150.f);
    src[2] = cv::Point2f(946.f, 561.f);
    src[3] = cv::Point2f(89.f, 720.f);

    dst[0] = cv::Point2f(0, 0);
    dst[1] = cv::Point2f(w - xDiff, 0);
    dst[2] = cv::Point2f(w - xDiff, h);
    dst[3] = cv::Point2f(0, h);

    cv::Mat transform = getPerspectiveTransform(src, dst);
    cv::Mat warpedImage;
    warpPerspective(originalImage, warpedImage, transform,
        cv::Size(cvRound(dst[2].x), cvRound(dst[2].y)), cv::INTER_AREA);

    cv::namedWindow("warpedImage:", 1);
    cv::imshow("warpedImage:", warpedImage);
    return warpedImage;
}

void showGaussImage(cv::Mat originalImage) {
    cv::Mat gauss;
    double sigmaStart = 0.0000000001;
    double sigmaEnd = 3.0;
    cv::Size size(5, 5);
    double step = (sigmaEnd - sigmaStart) / 100.0;
    cv::namedWindow("gauss:", 1);
    cv::GaussianBlur(originalImage, gauss, size, sigmaStart, sigmaStart);
    cv::imshow("gauss:", gauss);
    cv::imwrite("gauss/image000.bmp", gauss);
```

```
    while (cv::waitKey() != 27) {}

    for(int i = 1; i < 100; ++i){
        auto sigma = sigmaStart + step * i;
        cv::GaussianBlur(originalImage, gauss, size, sigma, sigma);
        cv::imshow("gauss:", gauss);


        std::stringstream str;
        str << "gauss/image" << std::setfill('0') << std::setw(3) << i << ".bmp";
        cv::imwrite(str.str(), gauss);

        cv::waitKey(35);
    }

    cv::destroyWindow("gauss:");
}
```