**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра ТВ**

отчет

**по лабораторной работе №2**

**по дисциплине «Интеллектуальные видеосистемы»**

Тема: разработка программы для морфологической фильтрации изображений

|  |  |  |
| --- | --- | --- |
| Студенты гр. 8571 |  | Темный Д.А.  Мальцева О.Н. |
| Преподаватель |  | Поздеев А.А. |

Санкт-Петербург

2023

**Цель работы**

Целью лабораторной работы является знакомство с методами морфологической фильтрации изображений. В соответствии с заданием на моделирование необходимо разработать программу в виде консольного приложения, реализующую следующие функции:

1. Загрузку растрового изображения, преобразование его в однотонное и вывод его на экран.
2. Преобразование изображения в бинарное и вывод его на экран.
3. Фильтрацию бинарного изображения морфологическими фильтрами, перечисленными в таблице 1.
4. Фильтрацию полутонового изображения морфологическими фильтрами, перечисленными в таблице 1.

Таблица 2.1. Задание на морфологическую фильтрацию

|  |  |
| --- | --- |
| **Вид изображения** | **Вид морфологического фильтра или оператора** |
| Черно-белое изображение | Фильтр бинарной эрозии |
| Фильтр бинарной дилатации |
| Оператор закрытия |
| Оператор открытия |
| Полутоновое изображение | Фильтр полутоновой эрозии |
| Фильтр полутоновой дилатации |
| Оператор закрытия |
| Оператор открытия |
| Оператор выделения контуров |
| Многомасштабный морфологический градиент |

**Теоретические сведения**

*Математическую морфологию* как системно-теоретический метод анализа изображений применяют в стандартных областях обработки/анализа цифровых изображений, таких как нелинейная фильтрация изображений; определение граней; подавление шумов; отображение форм, сглаживание и распознавание; скелетирование, кодирование. Алгоритм обработки изображения, на входе и выходе которого находятся *бинарные* изображения, называют двумерным системным фильтром. Алгоритм, который преобразует входное полутоновое в выходное *полутоновое* изображение, называют функциональным двумерным фильтром. Логические операции совершают между структурным элементом (маской) **S** и областью цифрового изображения **L**, выделенной этим структурным элементом. Результат операции помещают в новую битовую матрицу на место, где находится фокус маски. Используют структурные элементы различной формы и размерности (рис. 2.1).

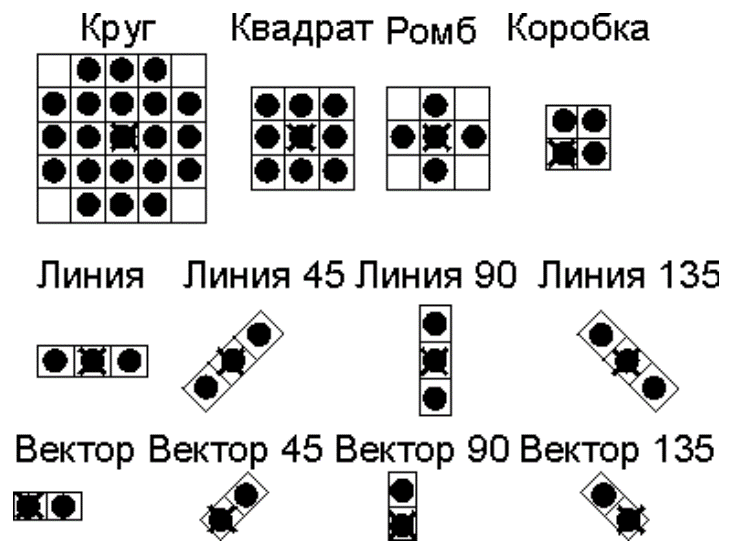


Рис. 2.1 – Структурные элементы на плоскости (крестом помечен фокус)

Рассмотрим основные операции морфологической фильтрации применительно к обработке бинарно квантованных изображений:

❖ Эрозию (*erosion*) обозначают символами **L**Θ**S** и осуществляют путем операции логического «И» между состоящей из единиц маской **S** размером и накрываемым маской фрагментом бинарно квантованного изображения **L**. По результатам логического умножения формируют новый массив изображения. Если накрываемый маской фрагмент содержит хотя бы один ноль, в элементе вновь формируемого изображения, соответствующем положению «фокуса» маски, устанавливают ноль. В результате этой операции все фрагменты исходного изображения по площади меньшие, чем маска, исчезают, а большие – «сжимаются» на величину маски.

❖ Наращивание (*dilation* - дилатация) обозначают символами и выполняют путем операции логического «ИЛИ»: в фокусе маски устанавливают единицу, если в накрываемом маской фрагменте находится хотя бы один единичный элемент. В результате изображение восстанавливают до исходных размеров.

❖ Открытие (*opening*) представляет собой комбинацию эрозии и наращивания, которые выполняют последовательно: ; в результате «открытия» элементы меньшие площади маски исчезают из изображения, а большие – восстанавливаются до своих исходных размеров

❖ Закрытие (*closing*) - порядок выполнения действий заменен на обратный: .

Открытие устраняет острые выступы и узкие перешейки в **L**, тогда как закрытие заполняет узкие заливы и малые отверстия. Форма и размер структурного элемента определяют природу и степень сглаживания. Если структурный элемент **S** имеет регулярную форму, открытие и закрытие можно рассматривать как нелинейную фильтрацию, которая сглаживает контуры входного сигнала.

Применительно к полутоновым изображениям операцию эрозии трактуют как нахождение минимума в пределах области исходного цифрового изображения, накрытой маской, и установку этого значения в результирующей матрице на место, где расположен фокус маски. В операции наращивания ищут максимум сигнала.

Морфологической фильтрации подвергают, в частности, одномерные полутоновые сигналы, при использовании структурного элемента S в виде линии или вектора (см. рис. 2.1). Эрозия (*erosion*) функции при использовании ограниченного структурного элемента расширяет минимумы , тогда как наращивание расширяет максимумы. Открытие с помощью S отрезает пики в сигнале, а закрытие заполняет спады, так что всегда для любых **L** и **S** (рис. 2.2).

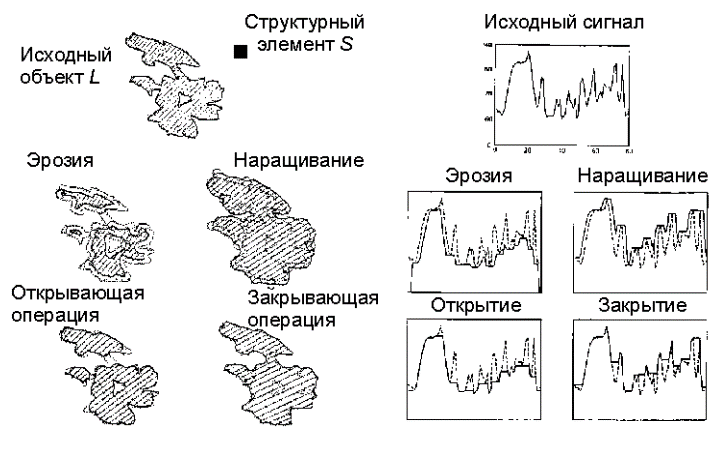


Рис. 2.2 – Результаты операций морфологической фильтрации применительно к бинарным (слева) и полутоновым изображениям (пунктиром показаны исходные объекты)

Препарирование изображений:

Видеоизображения препарируют с целью выделения информации, позволяющей наиболее эффективно решить задачи по обнаружению и сопровождению объектов. Пусть **S** начальный структурный элемент, один из представленных на рис. 2.1. Затем к нему *n* раз применяют операцию наращивания Если **S** выпуклый элемент, то в результате получают новый структурный элемент *n***S** той же формы, но большего размера. Если **S** - двумерный симметричный структурный элемент, то системная Разность (здесь минус в круге означает оператор эрозии) дает границу бинарного изображения **L**, а алгебраическая разность усиливает контуры полутонового изображения **L**. Величина *n* в *n***S** управляет толщиной контуров. Контуры различной ориентации получают с использованием одномерной структурной системы **S**, направленной должным образом. Более симметричной обработкой между изображением и задним планом является оценщик контуров , который аппроксимирует градиент функции **L**. Результаты извлечения полутонового контура с помощью морфологических операций показаны на рис. 2.3.



Рис. 2.3 – Контурный препарат

Если в кадре могут присутствовать объекты, как темнее фона, так и светлее фона, то для препарирования изображения используют *градиентный метод*. Напомним, что вектор-градиент составлен из первых частных производных в заданной точке изображения .

Чтобы выявить рельеф изображения используют норму вектора-градиента . В точках локальных минимумов и максимумов рельефа вектор-градиент равен нулю, следовательно, объекты, как темнее фона, так и светлее фона приводятся к одному и тому же (нулевому) уровню. Вычисления вектора-градиента для каждой точки кадра представляют достаточно трудоемкую задачу, поэтому используют приближение к норме вектора градиента – так называемый *многомасштабный морфологический градиент*. Пусть обозначает фильтрованное изображение, а – группу квадратных структурных элементов. Размер равен (2*i* + 1)(2*i* + 1) пикселей для , и обозначают наращивание и эрозию, соответственно. Многомасштабный морфологический

(2.1)

В соответствии с выражением (2.1) значения градиентов рассчитывают трижды с использованием структурных элементов различной размерности, а затем результаты складывают. Применим указанную методику обработки к видео кадру самолета, идущего на посадку (рис. 2.4, 2.5). Обращает на себя внимание тот факт, что как корпус самолета (он темнее фона), так и включенные фары, теперь отображены в виде темных участков, соответствующих минимальным значениям сигналов.



Рис. 2.4 – Видео кадр самолета

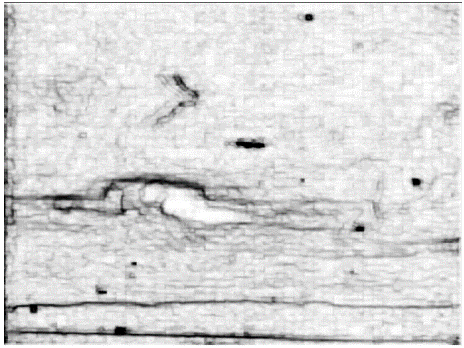


Рис. 2.5 – Карта морфологического градиента для изображения самолета

**Морфологическая фильтрация изображения**

Исходное изображение представлено на рис. 2.6, изображение в градациях серого на рис. 2.7.



Рис. 2.6 – Исходное изображение



Рис. 2.7 – Полутоновое исходное изображение

В качестве границы для генерации бинарного изображения используем среднее значение максимальной яркости пиксела (127), результат преобразования представлен на рис. 2.8.



Рис. 2.8 – Исходное бинарное изображение

**Обработка бинарного (черно-белого) изображения**

Для фильтров эрозии (рис. 2.9), дилатации (рис.2.10), а также для операторов открытия (рис. 2.11) и закрытия (рис. 2.12) использовался структурный элемент размером 3х3.



Рис. 2.9 – Результат применения фильтра бинарной эрозии



Рис. 2.10 – Результат применения фильтра бинарной дилатации



Рис. 2.11 – Результат применения оператора бинарного закрытия



Рис. 2.12 – Результат применения оператора бинарного открытия

**Обработка полутонового изображения**

При обработке полутонового изображения использовались те же структурные элементы и операторы, что и при обработке бинарного изображения (рис. 2.13 – 2.16)



Рис. 2. – Результат применения фильтра полутоновой эрозии



Рис. 2. – Результат применения фильтра полутоновой дилатации



Рис. 2. – Результат применения оператора полутонового закрытия



Рис. 2. – Результат применения оператора полутонового открытия

Для выделения контуров на изображении был вычислен модуль разности между результатами применения полутоновой дилатации и эрозии (рис. 2.17).



Рис. 2.17 – Результат применения оператора выделения контуров

Многомасштабный морфологический градиент представлен средним арифметическим (для каждого пиксела) между результатами применения оператора выделения контуров с размерами структурных элементов 3х3, 5х5 и 7х7 (рис. 2.18).



Рис. 2.18 – Результат применения многомасштабного морфологического градиента

**Вывод**

В ходе работы был исследован способ нелинейной фильтрации изображений: применение морфологических операторов – эрозию, дилатацию, открытие и закрытие – к бинарному (черно-белому) и полутоновому изображениям.

Эрозия позволяет убрать случайные шумы, относящиеся к выделяемому объекту (белые или более светлые пикселы), в частности сглаживая края объектов, что видно при применении к полутоновому изображению. Дилатация же позволяет проще различать объекты в затемненных частях изображения, однако в светлых областях может дать т.н. «пересвет».

Закрытие и открытие позволяет уменьшить недостатки эрозии и дилатации: излишних затемнений или пересветов на изображении, при сохранении возможности сглаживания границ объекта.

Также были рассмотрены способы получения контурного препарата: с использованием разницы результатов дилатации и эрозии и многомасштабного морфологического градиента. Оператор выделения контуров дает результат, визуально очень схожий с разностью гауссиан, однако обладает большей общей яркостью.

Результат применения многомасштабного морфологического градиента также визуально схож с использованием обобщенного оператора Собела, имеющий большую яркостью, контрастность и резкость границ.

**Листинг Кода**

**main.cpp:**

#include <iostream>

#include <string>

#include <opencv2/opencv.hpp>

#include <utils/utils.h>

#include <img\_to\_binary.h>

#include <morphological\_operations.h>

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

    std::string filename = "../kitten.png";

    std::string output\_folder = "output";

    cv::Mat image = cv::imread(filename, cv::IMREAD\_GRAYSCALE);

    cv::Mat binary\_img;

    cv::Mat binary\_erosion\_img;

    cv::Mat binary\_dilation\_img;

    cv::Mat binary\_opening\_img;

    cv::Mat binary\_closing\_img;

    cv::Mat grayscale\_erosion\_img;

    cv::Mat grayscale\_dilation\_img;

    cv::Mat grayscale\_opening\_img;

    cv::Mat grayscale\_closing\_img;

    cv::Mat outline\_img;

    cv::Mat gradient\_img;

    std::cout << "\nSave all? (0/1) ";

    int confirm = 0;

    std::cin >> confirm;

    if (confirm)

    {

        std::string create\_folder = "mkdir " + output\_folder;

        system(create\_folder.c\_str());

        output\_folder += "/";

        cv::imwrite(output\_folder + "grayscale.jpg", image);

        get\_binary(image, binary\_img);

        cv::imwrite(output\_folder + "binary\_img.jpg", binary\_img);

        erosion(binary\_img, binary\_erosion\_img);

        cv::imwrite(output\_folder + "binary\_erosion\_img.jpg", binary\_erosion\_img);

        dilation(binary\_img, binary\_dilation\_img);

        cv::imwrite(output\_folder + "binary\_dilation\_img.jpg", binary\_dilation\_img);

        opening(binary\_img, binary\_opening\_img);

        cv::imwrite(output\_folder + "binary\_opening\_img.jpg", binary\_opening\_img);

        closing(binary\_img, binary\_closing\_img);

        cv::imwrite(output\_folder + "binary\_closing\_img.jpg", binary\_closing\_img);

        erosion(image, grayscale\_erosion\_img);

        cv::imwrite(output\_folder + "grayscale\_erosion\_img.jpg", grayscale\_erosion\_img);

        dilation(image, grayscale\_dilation\_img);

        cv::imwrite(output\_folder + "grayscale\_dilation\_img.jpg", grayscale\_dilation\_img);

        opening(image, grayscale\_opening\_img);

        cv::imwrite(output\_folder + "grayscale\_opening\_img.jpg", grayscale\_opening\_img);

        closing(image, grayscale\_closing\_img);

        cv::imwrite(output\_folder + "grayscale\_closing\_img.jpg", grayscale\_closing\_img);

        outline(image, outline\_img);

        cv::imwrite(output\_folder + "outline\_img.jpg", outline\_img);

        gradient(image, gradient\_img);

        cv::imwrite(output\_folder + "gradient\_img.jpg", gradient\_img);

        std::cout << "\nShow all? (0/1) ";

        std::cin >> confirm;

        if (confirm)

        {

            show\_img("binary "+filename, binary\_img);

            show\_img("binary erosion "+filename, binary\_erosion\_img);

            show\_img("binary dilation "+filename, binary\_dilation\_img);

            show\_img("binary opening "+filename, binary\_opening\_img);

            show\_img("binary closing "+filename, binary\_closing\_img);

            show\_img("grayscale erosion "+filename, grayscale\_erosion\_img);

            show\_img("grayscale dilation "+filename, grayscale\_dilation\_img);

            show\_img("grayscale opening "+filename, grayscale\_opening\_img);

            show\_img("grayscale closing "+filename, grayscale\_closing\_img);

            show\_img("outline "+filename, outline\_img);

            show\_img("gradient "+filename, gradient\_img);

        }

    }

    std::cout << "\n";

    show\_img("input "+filename, image);

    if (confirm != 0)

        cv::waitKey(0);

}

**img\_to\_binary.h (конвертация полутонового изображения в бинарное):**

#include <opencv2/opencv.hpp>

void get\_binary(const cv::Mat &input\_img, cv::Mat &output\_img, int threshold = 255/2)

{

    output\_img = cv::Mat::zeros(input\_img.size(), CV\_8U);

    for ( int i = 0; i < input\_img.rows; i++ )

    {

        for ( int j = 0; j < input\_img.cols; j++ )

        {

            uchar pixel = input\_img.at<uchar>(i, j);

            output\_img.at<uchar>(i, j) = pixel < threshold ? 0 : 255;

        }

    }

}

**morphological\_operations.h (функции морфологических операторов):**

#include <vector>

#include <opencv2/opencv.hpp>

void erosion(const cv::Mat &input\_img, cv::Mat &output\_img, int aperture\_size = 3)

{

    output\_img = cv::Mat::zeros(input\_img.size(), CV\_8U);

    int frameWidth = aperture\_size/2;

    uchar min = 255;

    for ( int i = frameWidth; i < input\_img.rows - frameWidth; i++ )

    {

        for ( int j = frameWidth; j < input\_img.cols - frameWidth; j++ )

        {

            min = 255;

            for ( int ii = -frameWidth; ii <= frameWidth; ii++ )

            {

                for ( int jj = -frameWidth; jj <= frameWidth; jj++ )

                {

                    if ( input\_img.at<uchar>(i+ii, j+jj) < min )

                        min = input\_img.at<uchar>(i+ii, j+jj);

                }

            }

            output\_img.at<uchar>(i, j) = min;

        }

    }

}

void dilation(const cv::Mat &input\_img, cv::Mat &output\_img, int aperture\_size = 3)

{

    output\_img = cv::Mat::zeros(input\_img.size(), CV\_8U);

    int frameWidth = aperture\_size/2;

    uchar max = 0;

    for ( int i = frameWidth; i < input\_img.rows - frameWidth; i++ )

    {

        for ( int j = frameWidth; j < input\_img.cols - frameWidth; j++ )

        {

            max = 0;

            for ( int ii = -frameWidth; ii <= frameWidth; ii++ )

            {

                for ( int jj = -frameWidth; jj <= frameWidth; jj++ )

                {

                    if ( input\_img.at<uchar>(i+ii, j+jj) > max )

                        max = input\_img.at<uchar>(i+ii, j+jj);

                }

            }

            output\_img.at<uchar>(i, j) = max;

        }

    }

}

void opening(const cv::Mat &input\_img, cv::Mat &output\_img, int aperture\_size = 3)

{

    cv::Mat tmp\_img;

    erosion(input\_img, tmp\_img, aperture\_size);

    dilation(tmp\_img, output\_img, aperture\_size);

}

void closing(const cv::Mat &input\_img, cv::Mat &output\_img, int aperture\_size = 3)

{

    cv::Mat tmp\_img;

    dilation(input\_img, tmp\_img, aperture\_size);

    erosion(tmp\_img, output\_img, aperture\_size);

}

void outline(const cv::Mat &input\_img, cv::Mat &output\_img, int aperture\_size = 3)

{

    cv::Mat tmp\_img1;

    cv::Mat tmp\_img2;

    dilation(input\_img, tmp\_img1, aperture\_size);

    erosion(input\_img, tmp\_img2, aperture\_size);

    cv::absdiff(tmp\_img1, tmp\_img2, output\_img);

}

void gradient(const cv::Mat &input\_img, cv::Mat &output\_img)

{

    cv::Mat tmp\_img1;

    cv::Mat tmp\_img2;

    cv::Mat tmp\_img3;

    cv::Mat tmp\_img1\_c;

    cv::Mat tmp\_img2\_c;

    cv::Mat tmp\_img3\_c;

    cv::Mat output\_img\_c;

    outline(input\_img, tmp\_img1, 3);

    outline(input\_img, tmp\_img2, 5);

    outline(input\_img, tmp\_img3, 7);

    tmp\_img1.convertTo(tmp\_img1\_c, CV\_32F);

    tmp\_img2.convertTo(tmp\_img2\_c, CV\_32F);

    tmp\_img3.convertTo(tmp\_img3\_c, CV\_32F);

    output\_img\_c = (tmp\_img1\_c + tmp\_img2\_c + tmp\_img3\_c) / 3;

    output\_img\_c.convertTo(output\_img, CV\_8U);

}