минобрнауки россии

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербургский государственный электротехнический

университет «ЛЭТИ» им. В. И. Ульянова (Ленина)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Электронные методические указания**

**к выполнению лабораторных работ по дисциплине**

**«Основы компьютерного зрения»**

Учебно-методическое пособие

Санкт-Петербург

СПбГЭТУ «ЛЭТИ»

2024

Автор: С.В. Костичев

Электронные методические указания к выполнению лабораторных работ по дисциплине «Основы компьютерного зрения»: учебно-методическое пособие. СПбГЭТУ «ЛЭТИ», 2024. --- с.

Описывается цикл лабораторных работ.

.

Предназначено для студентов, обучающихся по направлению 09.03.01 «Информатика и вычислительная техника»

Одобрено

Методической комиссией факультета компьютерных технологий и

информатики

в качестве учебно-методического пособия

© СПбГЭТУ «ЛЭТИ», 2024

Оглавление

[Введение 4](#_Toc173414504)

[Лабораторная работа №1. 5](#_Toc173414505)

[БАЗОВЫЕ ОПЕРАЦИИ РИСОВАНИЯ ФИГУР НА ИЗОБРАЖЕНИИ 5](#_Toc173414506)

[1.1. Общие положения 5](#_Toc173414507)

[1.2. Предварительная подготовка к работе 7](#_Toc173414508)

[1.3. Порядок выполнения работы 7](#_Toc173414509)

[1.4. Содержание отчёта 8](#_Toc173414510)

[1.5. Контрольные вопросы 8](#_Toc173414511)

[Лабораторная работа №2. 9](#_Toc173414512)

[ГРАДАЦИОННЫЕ ПРЕОБРАЗОВАНИЯ И ЛОГИЧЕСКИЕ ОПЕРАЦИИ 9](#_Toc173414513)

[2.1. Общие положения 9](#_Toc173414514)

[2.2. Предварительная подготовка к работе 9](#_Toc173414515)

[2.3. Порядок выполнения работы 9](#_Toc173414516)

[2.4. Содержание отчёта 10](#_Toc173414517)

[2.5 Контрольные вопросы 11](#_Toc173414518)

[Лабораторная работа №3. 11](#_Toc173414519)

[НОРМАЛИЗАЦИЯ И ОПЕРАЦИИ НАД ГИСТОГРАММИ 11](#_Toc173414520)

[3.1. Общие положения 11](#_Toc173414521)

[3.2. Предварительная подготовка к работе 11](#_Toc173414522)

[3.3. Порядок выполнения работы 11](#_Toc173414523)

[3.4. Содержание отчёта 13](#_Toc173414524)

[3.5 Контрольные вопросы 13](#_Toc173414525)

[Лабораторная работа №4. 14](#_Toc173414526)

[ПРОСТРАНСТВЕННЫЕ ФИЛЬТРЫ 14](#_Toc173414527)

[4.1. Общие положения 14](#_Toc173414528)

[4.2. Предварительная подготовка к работе 14](#_Toc173414529)

[4.3. Порядок выполнения работы 14](#_Toc173414530)

[4.4. Содержание отчёта 18](#_Toc173414531)

[4.5. Контрольные вопросы 18](#_Toc173414532)

[Лабораторная работа №5. 18](#_Toc173414533)

[АФФИННЫЕ ПРЕОБРАЗОВАНИЯ, ГОМОГРАФИЯ И ПИРАМИДЫ ИЗОБРАЖЕНИЙ 18](#_Toc173414534)

[5.1. Общие положения 19](#_Toc173414535)

[5.2. Предварительная подготовка к работе 19](#_Toc173414536)

[5.3. Порядок выполнения работы 19](#_Toc173414537)

[5.4. Содержание отчёта 21](#_Toc173414538)

[5.5. Контрольные вопросы 21](#_Toc173414539)

[Лабораторная работа №6. 21](#_Toc173414540)

[МОРФОЛОГИЧЕСКИЕ ОПЕРАЦИИ 21](#_Toc173414541)

[6.1. Общие положения 22](#_Toc173414542)

[6.2. Предварительная подготовка к работе 22](#_Toc173414543)

[6.3. Порядок выполнения работы 22](#_Toc173414544)

[6.4. Содержание отчёта 25](#_Toc173414545)

[6.5. Контрольные вопросы 25](#_Toc173414546)

[Лабораторная работа №7. 25](#_Toc173414547)

[ЧАСТОТНЫЕ ФИЛЬТРЫ 25](#_Toc173414548)

[7.1. Общие положения 25](#_Toc173414549)

[7.2. Предварительная подготовка к работе 25](#_Toc173414550)

[7.3. Порядок выполнения работы 26](#_Toc173414551)

[7.4. Содержание отчёта 26](#_Toc173414552)

[7.5. Контрольные вопросы 27](#_Toc173414553)

[Лабораторная работа №8. 27](#_Toc173414554)

[ОБНАРУЖЕНИЕ КОНТУРНЫХ ПЕРЕПАДОВ, ВЫДЕЛЕНИЕ КОНТУРОВ, ПОИСК ПРЯМЫХ С ПОМОЩЬЮ ПРЕОБРАЗОВАНИЯ ХАФА 27](#_Toc173414555)

[8.1. Общие положения 27](#_Toc173414556)

[8.2. Предварительная подготовка к работе 28](#_Toc173414557)

[8.3. Порядок выполнения работы 28](#_Toc173414558)

[8.4. Содержание отчёта 32](#_Toc173414559)

[8.5. Контрольные вопросы 32](#_Toc173414560)

[Лабораторная работа №9. 33](#_Toc173414561)

[СЕГМЕНТАЦИЯ ИЗОБРАЖЕНИЙ 33](#_Toc173414562)

[9.1. Общие положения 33](#_Toc173414563)

[9.2. Предварительная подготовка к работе 33](#_Toc173414564)

[9.3. Порядок выполнения работы 33](#_Toc173414565)

[9.4. Содержание отчёта 35](#_Toc173414566)

[9.5. Контрольные вопросы 35](#_Toc173414567)

[Лабораторная работа №10. 35](#_Toc173414568)

[ОПРЕДЕЛЕНИЕ ОБЪЕКТА С ПОМОЩЬЮ ДЕТЕКТОРОВ И КЛАССИФИКАТОРОВ 35](#_Toc173414569)

[10.1. Общие положения 35](#_Toc173414570)

[10.2. Предварительная подготовка к работе 36](#_Toc173414571)

[10.3. Порядок выполнения работы 37](#_Toc173414572)

[10.4. Содержание отчёта 38](#_Toc173414573)

[10.5. Контрольные вопросы 38](#_Toc173414574)

[Приложения. 39](#_Toc173414575)

[1. УСТАНОВКА БИБЛИОТЕКИ OpenCV В СРЕДЕ MICROSOFT VS 39](#_Toc173414576)

[2. УСТАНОВКА OpenCV-Python НА Windows 42](#_Toc173414577)

[3. ИСПОЛЬЗУЕМЫЕ ИЗОБРАЖЕНИЯ 42](#_Toc173414578)

# Введение

Методические указания к лабораторным работам предназначены для ознакомления студентами с возможностями библиотеки OpenCV (Open Source Computer Vision Library). OpenCV является одной из наиболее популярных библиотек компьютерного зрения с открытыми исходными кодами, в состав которой входит большое количество функций обработки изображений и видео в реальном времени. Библиотека реализована на языках C/C++, поддерживает множество языков программирования, включая C++, Python, Java и другие. Характерной особенностью OpenCV является модульность архитектуры. Наиболее используемыми являются следующие модули:

* core – модуль, содержащий объявление всех структур данных, включая базовую структуру для представления многомерного массива Mat и функции работы с ней.
* imgproc – модуль обработки изображений, который включает в себя линейную и нелинейную фильтрацию, геометрические преобразования изображений (в частности, масштабирование), преобразования цветовых пространств и т.д.
* highgui – модуль, позволяющий отображать рабочие изображения, проигрывать видео и создавать простые интерфейсы управления.
* ml – модуль, содержащий реализацию некоторых алгоритмов машинного обучения.
* objdetect – модуль детектирования объектов.
* video – модуль анализа видео, включающий функции оценивания движения на видео, вычитания фона и слежения за объектами на последовательности кадров видеопотока.
* features2d – модуль выделения и сопоставления особых точек на изображениях.

Более детально описание модулей [**https://docs.opencv.org/4.8.0/modules.html**](https://docs.opencv.org/4.8.0/modules.html) **.**

Указания содержат задания, рекомендации по выполнению лабораторных работ и содержанию отчётов.

В приложении 1 приведена последовательность шагов при установке библиотеки OpenCV в среде Microsoft Visual Studio.

В приложении 2 приведена последовательность шагов при установке библиотеки OpenCV-Python на Windows

Установка OpenCV на C++ длительна и сложна, на Python установка библиотеки проста.

В лабораторных работах предполагается использование языка Python с библиотекой OpenCV.

Первое, что сделать при создании программы — это импортировать библиотеки.

*Python:*

import cv2 # собственно OpenCV

import numpy as np # для работы с математикой

import matplotlib.pyplot as plt # для вывода картинки

Используемые изображения представлены в приложении 3.

# Лабораторная работа №1.

# БАЗОВЫЕ ОПЕРАЦИИ РИСОВАНИЯ ФИГУР НА ИЗОБРАЖЕНИИ

Цель работы: изучить применение базовых операций рисования на изображении

## 1.1. Общие положения

В OpenCV любое изображение представляет собой матрицу интенсивностей.

В *Python* это NumPy-массив, отличающийся от обычных списков и кортежей в тем, что должен состоять только из элементов одного типа.

Чтение, отображение и запись изображений являются базовыми для обработки изображений и компьютерного зрения.

**Чтение изображения из файла**

Для чтения изображения используйте функцию imread() в OpenCV:

imread( filename[, flags]) -> retval

Параметры:

*filename -* имя загружаемого файла изображения.

*flags -* необязательный флаг, позволяющий указать, как должно быть представлено изображение. Наиболее распространенные:

*- cv2.IMREAD\_UNCHANGED или -1*

*- cv2.IMREAD\_GRAYSCALE или 0*

*- cv2.IMREAD\_COLOR или 1 (по умолчанию)*

*retval* - если картинка не загрузилась (ошибка в имени/пути, битое изображение), в retval запишется None.

*Важно:* OpenCV считывает цветные изображения в формате BGR

Примеры:

img\_color = cv2.imread('test.jpg',cv2.IMREAD\_COLOR)

Для облегчения анализа изображения картинку делают черно-белой:

img = cv2.imread('MyPhoto.jpg', cv2.IMREAD\_GRAYSCALE)

**Отображение изображения**

Для отображения изображения используйте функцию imshow() в OpenCV:

imshow(window\_name, image) -> None

Параметры:

*window\_name -* название окна, в котором будет изображение*.*

*image* - изображение, которое вы хотите отобразить*.*

Чтобы отобразить несколько изображений одновременно, укажите новое имя окна для каждого изображения, которое вы хотите отобразить.

Функция imshow() предназначена для использования вместе с функциями waitKey() и destroyAllWindows() / DestroyWindow().

Функция waitKey(t) является функцией привязки к клавиатуре.

*t* - время (в миллисекундах), в течение которого будет отображаться окно. Если t=0, программа ожидает нажатия клавиши неопределенное время, иначе ожидает время t.

Функция destroyAllWindows()уничтожает все созданные нами окна. Если нужно уничтожить конкретное окно, укажите точное имя окна в качестве аргумента.

Пример:

cv2.imshow('grayscale image',img\_grayscale)

cv2.waitKey(0)

cv2.destroyAllwindows()

**Запись изображения в файл**

Для записи изображения в файл используйте функцию imwrite() в OpenCV:

imwrite(filename, img[, params]) -> retval

Параметры:

*filename* - имя файла, которое должно включать расширение имени файла (например .png, .jpg и т.д.).

*img* - изображение, которое вы хотите сохранить.

*params -* параметры, зависящие от формата, закодированные в виде пар

Функция возвращает значение True, если изображение успешно сохранено.

Пример:

cv2.imwrite('grayscale.jpg',img\_grayscale)

Рисование на изображение осуществляется с помощью функций line, rectangle, circle, ellipse, fillPoly, polylines, putText.

## 1.2. Предварительная подготовка к работе

Ознакомиться с базовыми операциями рисования фигур в библиотеке OpenCV.

## 1.3. Порядок выполнения работы

1. Создать в качестве изображения черный квадрат размером 400\*400 пикселей.

2. Нарисовать в нем отрезок зеленого цвета между точками с координатами (15, 20) и (70, 50).

3. Нарисовать в нем окружность синего цвета с центром в точке с координатами (200, 200) и радиусом 32

4. Нарисовать в нем эллипс красного цвета со следующими параметрами

- центр в точке с координатами (200, 200) ,

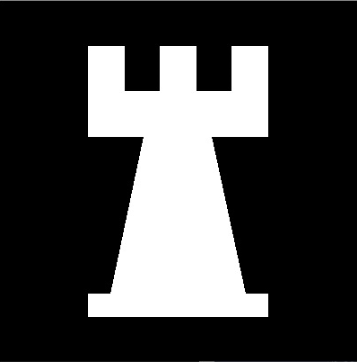
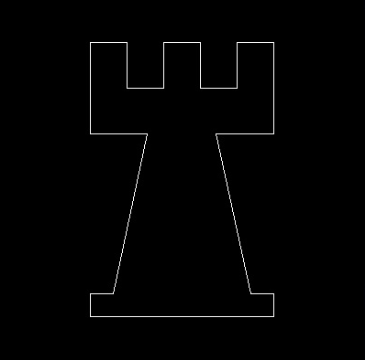
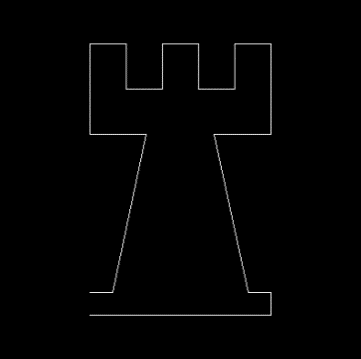
- оси 100 и 160

- угол поворота эллипса в градусах 45

- начальный и конечный углы эллиптической дуги в градусах 0 и 360

5. Нарисовать в нем прямоугольник желтого цвета с координатами противоположных вершин (15, 20) и (70, 50)

6. Нарисовать в нем полигоны следующих видов

[210, 390] – координаты точки, указанной стрелкой

При реализации используйте следующие функции:

**fillPoly**(img, pts, color) -> img

Параметры

*img* Image.

*pts* Array of polygons where each polygon is represented as an array of points.

*color* Polygon color.

**polylines**(img, pts, isClosed, color) -> img

Параметры

*img* Image.

*pts* Array of polygonal curves.

*isClosed* Флажок, указывающий, замкнуты ли нарисованные полилинии или нет. Если они замкнуты, функция проводит линию от последней вершины кривой к ее первой вершине.

*color* Polyline color.

7. Нарисовать в нем текстовую строку “Hello, world!”

- шрифтом ITALIC пурпурного цвета (координаты нижнего левого угла (20, 375))

- шрифтом HERSHEY\_SCRIPT\_COMPLEX голубого цвета (координаты нижнего левого угла (80, 50))

При реализации используйте функцию

**putText**(img, text, org, fontFace, fontScale, color[, thickness]) -> img

Параметры

*img* Image.

*text* Text string to be drawn.

*org* Нижний левый угол текстовой строки на изображении.

*fontFace* Font type, see HersheyFonts.

*fontScale* Font scale factor that is multiplied by the font-specific base size.

*color* Text color.

*thickness* Thickness of the lines used to draw a text

## 1.4. Содержание отчёта

Отчет по лабораторной работе должен содержать:

- титульный лист;

- задание на лабораторную работу;

- программное и аппаратное окружение при выполнении работы, включая версии Python, библиотеки OpenCV

- тексты программ;  
 - примеры запуска программ;

## 1.5. Контрольные вопросы

1. В чем отличие использования функций fillPoly и polylines?

2. Как сделать полигон синего цвета?

# Лабораторная работа №2.

# ГРАДАЦИОННЫЕ ПРЕОБРАЗОВАНИЯ И ЛОГИЧЕСКИЕ ОПЕРАЦИИ

Цель работы: изучить применение логарифмического, степенного и кусочно-линейного преобразований, логических операций

## 2.1. Общие положения

Градационные преобразования (логарифмическое, степенное и кусочно-линейное) являются поэлементными преобразованияминад одним изображением.

Логарифмическое преобразование используется для отображенияширокого диапазона больших входных значений в узкий диапазон выходных значений илиузкого диапазона малых входных значений в широкий диапазон выходных значений. Для этих же целей более универсальными являются степенные преобразования (гамма-коррекция).

Кусочно-линейное преобразование используется для усиления контраста.

Логические операции являются поэлементными преобразованияминад двумя и более изображениями (исключение NOT). Операции AND и ORиспользуются для маскирования(выделения части изображения). Операция NOTотносится к функциям градационного преобразования.

## 2.2. Предварительная подготовка к работе

1. Ознакомиться с градационными преобразованиями

2. Ознакомиться с логическими операциями над изображениями: инверсия, конъюнкция, дизъюнкция, исключающее ИЛИ.

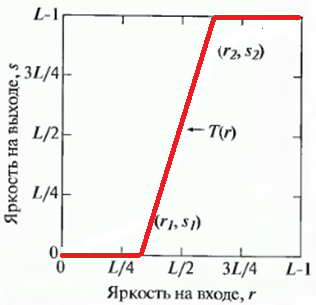
## 2.3. Порядок выполнения работы

1. Градационные преобразования

- выполнить логарифмическое преобразование изображения moscow.jpg.

- выполнить гамма-коррекцию изображения moscow.jpg для следующих значений гамма: 0.1, 0.5, 1.2, 2.2

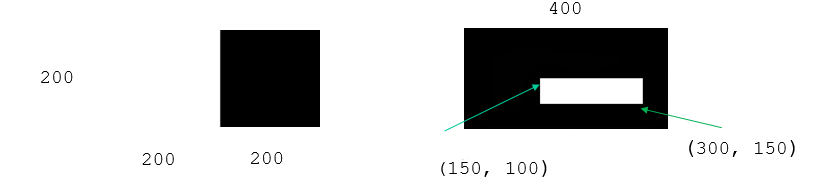
- выполнить над изображением moscow.jpg кусочно-линейное преобразование вида



с параметрами r1 = 70, s1 = 0, r2 = 140, s2 = 255

2. Логические операции

- создать 2 изображения img1 и img2 размером 200х400 вида



img1 img2

И записать их в файлы drawing1.jpg и drawing2.jpg соответственно

- выполнить логические операции: инверсия, конъюнкция, дизъюнкция, исключающее ИЛИ над изображениями img1 и img2.

Напоминание: Результат выполнения операции XOR над двумя битами равен 1, когда один из битов равен 1

- выполнить отображения исходного и результирующего изображений для всех логических операций

## 2.4. Содержание отчёта

Отчет по лабораторной работе должен содержать:

- титульный лист;

- задание на лабораторную работу;

- программное и аппаратное окружение при выполнении работы, включая версии Python, библиотеки OpenCV

- тексты программ;  
- результаты запуска программ;

## 2.5 Контрольные вопросы

1. Назначение логарифмических и степенных преобразований

2. График бинарного (порогового) преобразования

3. Назовите логические операции над изображениями.

# Лабораторная работа №3.

# НОРМАЛИЗАЦИЯ И ОПЕРАЦИИ НАД ГИСТОГРАММИ

Цель работы: изучить применение нормализации над изображениями и базовых функций OpenCV для работы с гистограммами

## 3.1. Общие положения

Нормализация изображения может улучшить их качество и сделать их более пригодными для анализа.

Слабый контраст – распространенный дефект изображений и кадров видео. Существуют следующие основные методы повышения контраста изображения: *линейная растяжка гистограммы* (линейное контрастирование), *эквализация* (equalization или линеаризация) гистограммы. Наиболее эффективный подход к решению задачи повышения контраста - эквализация гистограммы.

Эквализация гистограммы автоматически находит функцию преобразования, которая стремиться сформировать выходное изображение с равномерной гистограммой.

## 3.2. Предварительная подготовка к работе

1. Ознакомиться с операцией нормализации изображений.

2. Ознакомиться со следующими операциями над гистограммами в библиотеке OpenCV: вычисление и эквализация.

3. Ознакомиться с библиотекой Matplotlib для решения задач, возникающих при работе с графиками.

## 3.3. Порядок выполнения работы

1. Нормализация изображения и построение гистограмм исходного и нормализованного изображений

Выполните:

* загрузку изображения image3.jpg и его отображение
* преобразование изображения в оттенки серого и его отображение
* постройте и отобразите гистограмму серого
* нормализация типа MINMAX серого и его отображение. Выберите значения нижней и верхней границ диапазона яркостей 63 и 255 соответственно
* постройте и отобразите гистограмму нормализованного серого

При реализации используйте функции

**normalize**(src, dst[, alpha[, beta[, norm\_type[, dtype[, mask]]]]]) -> dst

Параметры:

*src* - входной массив.

*dst* - выходной массив того же размера, что и src.

*alpha* - значение нормы для нормализации или нижняя граница диапазона в случае нормализации диапазона.

*beta* - верхняя граница диапазона при нормировке диапазона; не используется для нормализации нормы.

*normType* - тип нормализации (NORM\_MINMAX, NORM\_INF, NORM\_L1, NORM\_L2).

*dtype* - при отрицательном значении выходной массив имеет тот же тип, что и *src*;

*mask* - необязательная маска операции (может указывать часть входного массива для нормализации).

По дефолту *alpha=1, beta=0, norm\_type=NORM\_L2*

Обычно используют *norm\_type=NORM\_MINMAX и задают* нижнюю и верхнюю границы диапазона

**calcHist**(images, channels, mask, histSize, ranges[, hist[, accumulate]]) -> hist

Параметры:

*images* - список изображений в виде массивов numpy. Все изображения должны быть одного типа и размера.

*channels* - список каналов, используемых для вычисления гистограммы. Число каналов изменяется от 0 до 2 (0- blue, 1- green, 2- red).

*mask* - необязательная маска, показывающая, какие пиксели учитывать при вычислении гистограммы.

*histSize:* размеры гистограммы в каждом измерении*.*

*ranges* - массив массивов, описывающих границы интервалов гистограммы по каждому измерению.

*hist* - результирующая гистограмма

*accumulate*=false - по умолчанию этот параметр равен false. Он определяет, является ли гистограмма кумулятивной (накопительная идентификация). Позволяет накапливать простую гистограмму по нескольким изображениям или обновлять гистограмму во времени.

3. Вычисление и эквализация гистограммы (на примере простого изображения)

Выполните:

* загрузку изображения color4.jpg (956х279пикс). и его отображение. Изображение содержит 4 квадрата по 66681пикс (956х279:4) с яркостями 0,1,2 и 3.
* преобразование изображения в оттенки серого и его отображение
* постройте и отобразите гистограмму серого
* эквализация серого и отображение полученного изображения
* постройте и отобразите гистограмму изображения с эквализацией

Объясните визуальную разницу серого и результирующего изображений.

При реализации используйте функцию

**equalizeHist**(src[, dst]) -> dst

Параметры:

*src* - исходное 8-битное одноканальное изображение.

*dst* - целевое изображение того же размера и типа, что и src.

4. Вычисление, нормализация и эквализация гистограммы (на реальном изображении)

Выполните:

* загрузку изображения image\_gray\_63-192.jpg и его отображение (диапазон яркости 63-192).
* преобразование изображения в оттенки серого и его отображение
* постройте и отобразите гистограмму серого
* нормализация типа MINMAX серого и его отображение. Выберите значения нижней и верхней границ диапазона яркостей 0 и 255 соответственно
* постройте и отобразите гистограмму нормализованного изображения
* эквализация серого и отображение полученного изображения
* постройте и отобразите гистограмму изображения с эквализацией

## 3.4. Содержание отчёта

Отчет по лабораторной работе должен содержать:

- титульный лист;

- задание на лабораторную работу;

- программное и аппаратное окружение при выполнении работы, включая версии Python, библиотеки OpenCV

- тексты программ;  
- результаты запуска программ;

## 3.5 Контрольные вопросы

1. Понятие гистограммы. Вероятностная трактовка гистограммы.

2. Суть метода эквализации гистограммы.

3. Отличие нормализации и эквализации

4. Какой метод можно использовать чтобы задать желаемую форму гистограммы?

# Лабораторная работа №4.

# ПРОСТРАНСТВЕННЫЕ ФИЛЬТРЫ

Цель работы: Изучить операции сглаживания изображений (низкочастотные фильтры) и повышения резкости изображений (высокочастотные фильтры)

## 4.1. Общие положения

*Предобработка* подразумевает преобразование исходного изображения в некоторое новое изображение.

Методы обработки в **пространственной** области (пространственные методы) объединяют подходы, основанные на прямом манипулировании пикселями изображения.

Наиболее популярным способом обработки является *фильтрация*, которая в большинстве приложений используется для удаления шумов. Результат фильтрации – изображение того же размера, что и исходное, но содержащее значения интенсивностей пикселей, обновленные в соответствии с некоторым правилом.

Для операции сглаживания используются фильтры низкой частоты (ФНЧ), для повышения резкости – фильтры высокой частоты (ФНЧ)

## 4.2. Предварительная подготовка к работе

1. Ознакомиться со сглаживающими пространственными фильтрами:

Гаусса, усредняющими, медианным.

2. Ознакомиться с фильтрами повышения резкости:

Превитта, Собеля, Шарра и Лапласса

## 4.3. Порядок выполнения работы

1. Фильтрация изображения с помощью произвольного линейного фильтра (заданного пользователем) 3х3.

Разработайте программу, обеспечивающую загрузку изображения image0.jpg и применение линейного фильтра с вещественным ядром 3х3, заданным константой

kernel = {-0.1, 0.2, -0.1,

0.2, 3.0, 0.2,

-0.1, 0.2, -0.1};

Выполните отображение исходного и результирующего изображений.

При реализации используйте функцию

filter2D (src, ddepth, kernel[, dst[, anchor[, delta[, borderType]]]]) -> dst

Параметры:

*src* - входное изображение;.

*dst* - целевое изображение того же размера, что и *src*.

*ddepth* - задает глубину выходного изображения; обычно = -1 означает, что нужно использовать такую же глубину, как у входного

*kernel* - ядро свертки (а точнее ядро корреляции); матрица чисел с плавающей запятой;

*anchor -* определяет положение так называемого якорного пикселя. По умолчанию значение (–1, –1) означает, что якорь находится в центре ядра

*delta* - необязательное значение, добавляемое к отфильтрованным пикселям перед их сохранением в dst.

*borderType* - определяет поведение на краях изображения. default = BORDER\_DEFAULT= gfedcb|abcdefgh|gfedcba

1. Фильтрация изображения с помощью линейного арифметического взвешенного усредняющего фильтра (ФНЧ) 5х5, заданного пользователем.

Постройте арифметический взвешенный усредняющий фильтр (ФНЧ) с маской 5х5. Вкачестве меры расстояния используйте расстояние Евклида.

Элементы маски формируются по формуле

*Mask[i, j] = (maxDist - Dist[i, j]) / maxDist,* где *maxDist=√8*

Разработайте программу, обеспечивающую загрузку изображения image0.jpg и применение фильтра с полученным ядром

Выполните отображение исходного и результирующего изображений.

При реализации используйте функцию *filter2D.*

1. Фильтрация изображения с помощью линейного арифметического однородного усредняющего фильтра (функция boxFilter) и фильтра Гаусса (функция GaussianBlur)

Разработайте программу, обеспечивающую загрузку изображения image0.jpg и применение фильтров с вещественным ядром 5х5.

Для boxFilter выполнить для 2-х значений параметра normalize.

Выполнить отображение исходного и результирующего изображений

При реализации используйте функции

**boxFilter**(src, ddepth, ksize[, dst[, anchor[, normalize[, borderType]]]]) -> dst

Параметры:

*src* - входное изображение.

*dst* - целевое изображение того же размера, что и *src*.

*ddepth* - задает глубину выходного изображения; –1 означает, что нужно использовать такую же глубину, как у входного

*ksize* - размер ядра.

*anchor -* определяет положение так называемого якорного пикселя. По умолчанию значение (–1, –1) означает, что якорь находится в центре ядра

*normalize* - если *normalize=True*, то каждый выходной пиксель вычисляется как среднее арифметическое соседей с коэффициентами ядра 1/n, где n – число соседей. Если *normalize=False*, то все коэффициенты равны 1.

default = *True*

*borderType* - определяет поведение на краях изображения. default = BORDER\_DEFAULT= gfedcb|abcdefgh|gfedcba

**GaussianBlur**(src, ksize, sigmaX[, dst[,sigmaY[,borderType]]]) -> dst

Параметры:

*src* - входное изображение.

*dst* - целевое изображение того же размера, что и *src*.

*ksize* - размер ядра по Гауссу. *ksize.width* и *ksize.height* могут различаться, но оба они должны быть положительными и нечетными.

*sigmaX* - стандартное отклонение ядра Гаусса в направлении X (сигма нормального распределения в направлении X); если *sigmaX=0*, то стандартные отклонения вычисляются исходя из размера ядра

*sigmaY* - стандартное отклонение ядра Гаусса в направлении Y; если sigmaY =0, он устанавливается равным sigmaX;

*borderType* - определяет поведение на краях изображения. default = BORDER\_DEFAULT = gfedcb|abcdefgh|gfedcba

1. Фильтрация изображения с помощью нелинейных медианного (функцияmedianBlur**)** и двунаправленного фильтров (функцияbilateralFilter**)**

Разработайте программу, обеспечивающую загрузку изображения image0.jpg и применение фильтров

- медианного с линейным размером апертуры 5

- двустороннего с sigmaColor=10, sigmaSpace=5.

Выполнить отображение исходного и результирующего изображений

При реализации используйте функции

**medianBlur**(src, ksize[,dst]) -> dst

Параметры:

*src* - входное изображение.

*dst* - целевое изображение того же размера, что и *src*.

ksize - линейный размер апертуры (ядра); он должен быть нечетным и больше единицы, например: 3, 5, 7.

**bilateralFilter**(src, d, sigmaColor, sigmaSpace[, dst[, borderType]]) -> dst

Параметры:

*src* - входное изображение.

*dst* - целевое изображение того же размера, что и *src*.

*d* − диаметр окрестности пикселя. Если не задавать *d* явно, а положить равным –1, то функция автоматически вычислит значение,

исходя из *sigmaSpace.*

*sigmaColor* − определяет одномерное гауссово распределение, которое определяет степень допустимости различий в интенсивности пикселей.

*sigmaSpace* − определяет пространственную протяженность ядра как в направлении x, так и в направлении y ( как фильтр Гаусса). Малые значения *sigmaSpace* (10) дают слабый, но заметный эффект, а большие (150) – очень сильный эффект, но изображение приобретает «мультяшный» вид.

*borderType* - определяет поведение на краях изображения. default = BORDER\_DEFAULT = gfedcb|abcdefgh|gfedcba

1. Фильтрация изображения с помощью фильтров повышения резкости

Разработать программу, обеспечивающую загрузку изображения cat.jpg и применение фильтров Превитта, Собеля, Шарра, Лапласса и модифицированного фильтра Лапласса для повышения резкости.

Выполнить отображение исходного и результирующего изображений

При реализации используйте функции

**filter2D** для реализации фильтра Превитта и модифицированного фильтра Лапласса

**Sobel**(src, ddepth, dx, dy[, dst[, ksize[, scale[, delta[, borderType]]]]]) -> dst

Параметры:

*src* - входное изображение.

*dst* - выходное изображение того же размера, что и src.

*ddepth* - глубина выходного изображения;

*dx* - порядок производной в направлении x; обычно 1 или 0

*dy* - порядок производной в направлении y; обычно 1 или 0

*ksize* - размер ядра Собела: 1,3,5,7; default = 3

*scale* - необязательный масштабирующий коэффициент; default = 1

*delta* - необязательное значение, прибавляемое к результирующим пикселям перед сохранением *в dst*; default = 0

borderType - определяет поведение на краях изображения; default = BORDER\_DEFAULT

В случае *ksize=FILTER\_SCHARR* оператор Собела = оператор Шарра

**Scharr**(src, ddepth, dx, dy[, dst[, scale[, delta[, borderType]]]]) -> dst

Параметры:

*src* - входное изображение.

*dst* - выходное изображение того же размера, что и src.

*ddepth* - глубина выходного изображения;

*dx* - порядок производной в направлении x; обычно 1 или 0

*dy* - порядок производной в направлении y; обычно 1 или 0

*scale* - необязательный масштабирующий коэффициент; default = 1

*delta* - необязательное значение, прибавляемое к результирующим пикселям перед сохранением *в dst*; default = 0

borderType - определяет поведение на краях изображения ; default = BORDER\_DEFAULT

В операторе Шарра размер ядра всегда =3

**Laplacian**(src, ddepth[, dst[, ksize[, scale[, delta[, borderType]]]]]) -> dst

Параметры:

*src -* исходное изображение*.*

*dst -* целевое изображение того же размера, что и *src.*

*ddepth -* желаемая глубина конечного изображения (–1 означает, что нужно использовать такую же глубину, как у входного).

*ksize -* размер апертуры, используемый для вычисления фильтров 2-ой производной, является положительным четным числом. (по умолчанию *ksize* =1 и применяется апертура размером 3x3 и используется маска, изотропная для поворотов на углы, кратные pi/2) *.*

*scale -* необязательный масштабирующий коэффициент; default = 1

*delta -* необязательное значение, прибавляемое к результирующим пикселям перед сохранением *в dst*; default = 0

*borderType -* определяет поведение на краях изображения; default = BORDER\_DEFAULT

## 4.4. Содержание отчёта

Отчет по лабораторной работе должен содержать:

- титульный лист;

- задание на лабораторную работу;

- программное и аппаратное окружение при выполнении работы, включая версии Python, библиотеки OpenCV

- тексты программ;

- результаты запуска программ

## 4.5. Контрольные вопросы

1. Дать определения понятиям фильтрации, маски, ядра, окна фильтра

2. Приведите пример нелинейной фильтрации

3. К каким последствиям приводит применение градиентных пространственных фильтров?

4. Какие фильтры можно использовать для удаления шума?

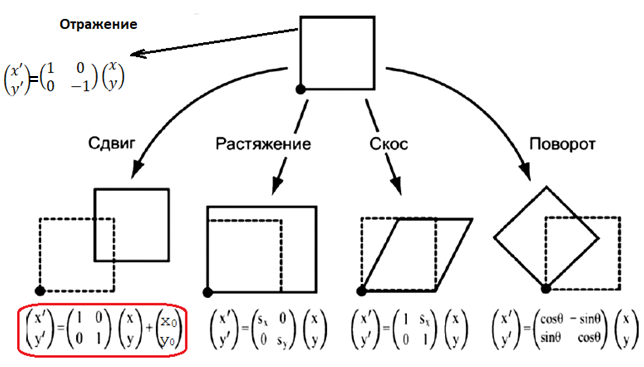
# Лабораторная работа №5.

# АФФИННЫЕ ПРЕОБРАЗОВАНИЯ, ГОМОГРАФИЯ И ПИРАМИДЫ ИЗОБРАЖЕНИЙ

Цель работы: Изучить аффинные преобразования

## 5.1. Общие положения

К основным типам *аффинных* преобразований относятся сдвиг, масштабирование (растяжение), поворот, скос, отражение



## 5.2. Предварительная подготовка к работе

Ознакомиться с основными типами аффинных преобразованийи представлением их с помощью умножения матриц.

Ознакомиться с гомографией и возможностями ее использования.

Ознакомиться с пирамидами изображений и возможностями их построения вниз и вверх.

## 5.3. Порядок выполнения работы

1. Реализация аффинных преобразований

В качестве исходного изображения возьмите cat.jpg

* 1. Выполните операцию поворота изображения на 45 градусов с коэффициентом масштабирования 1. В качестве центра вращения использовать центральную точку изображения. Для наглядности размер выходного изображения увеличить вдвое
  2. Выполните операцию растяжения (масштабирования) – уменьшить размер по ширине в 2 раза. Выполнить 2 варианта:

- использовать функцию resize

- использовать матрицу преобразования

* 1. Выполните операцию сдвига изображения на (50, 50). Для наглядности размер выходного изображения увеличить вдвое
  2. Выполните операцию отражения относительно оси 0X и 0Y
  3. Выполните операцию скоса относительно оси 0X и 0Y с коэффициентом скоса 0.2. ). Для наглядности размер выходного изображения увеличить вдвое.

При реализации аффинных преобразований используйте следующие функции:

**warpAffine**(src, M, dsize[, dst[, flags[, borderMode[, borderValue]]]]) -> dst

Параметры:

*src* - входное изображение.

*dst* - выходное изображение

*M -* **матрица преобразования** src в dst (**2х3**)

*dsize* - размер выходного изображения

*flags -* метод интерполяции

◦ *INTER\_NEAREST*: интерполяция по ближайшему соседу;

◦ *INTER\_LINEAR*: билинейная интерполяция, подразумевается по умолчанию;

*Отличие методов интерполяции: в* интерполяции по ближайшему соседу в расчет принимается значение яркости в одной ближайшей точке, а в билинейной – в 4-х ближайших точках

*borderMode – метод* экстраполяции пикселей (см. *borderInterpolate ()*);

*borderValue –* используется, если *borderMode= BORDER\_CONSTANT;* по умолчанию это 0.

Для создания матрицы *M* преобразования src в dst размером 2х3 используется функция

**getRotationMatrix2D**(center, angle, scale) -> retval

Параметры:

*center* – определяет центр вращения,

*angle* – угол поворота (в направлении против часовой стрелки),

*scale* – коэффициент масштабирования.

**warpPerspective**(src, M, dsize[, dst[, flags[, borderMode[, borderValue]]]]) -> dst

Параметры функции аналогичны warpAffine, только

*M* - **матрица преобразования** src в dst (**3х3**)

**resize**(src, dsize[, dst[, fx[, fy[, interpolation]]]]) -> dst

Параметры:

*src* - входное изображение.

*dst* - выходное изображение

*dsize* - новый размер; если *dsize#0, то fx = 0 и fy = 0*

*fx, fy* – коэффициенты масштабирования. По умолчанию =0

*interpolation -* метод интерполяции

* *INTER\_NEAREST*: интерполяция по ближайшему соседу (в качестве выходного пикселя берется ближайший к нему входной пиксель); используется для *увеличения* изображения
* *INTER\_AREA:* использует отношение площади пикселя для повторной выборки; подходит для *уменьшения* размера изображения
* *INTER\_LINEAR*: билинейная интерполяция, подразумевается по умолчанию (вычисляется взвешенная сумма соседних пикселей с весами, зависящими от расстояния до выходного);

1. Гомография

2.1 Пусть даны 2 изображения книги: book2.jpg и book1.jpg

Примените гомографию к изображению book2, чтобы книга на этом изображении стала выровненной с книгой на изображении book1

* 1. Пусть есть изображение книги: book1.jpg.

Примените гомографию к изображению book1, чтобы получить обложку книги

2.3 Пусть есть изображение Таймс-сквер times-square.jpg и нового рекламного щита first-image.jpg

Примените гомографию к изображению times-square.jpg, чтобы установить новую рекламу first-image.jpg на щит

При реализации гомографии используйте следующие функции

**findHomography**(srcPoints, dstPoints) -> retval, mask

Параметры:

*srcPoints* - массив точек в исходном изображении

*dstPoints* -массивы точек в выходном изображении

*retval* - гомография

*mask* - необязательная маска вывода

**warpPerspective**(src, M, dsize[, dst[, flags[, borderMode[, borderValue]]]]) -> dst

Параметры функции аналогичны warpAffine, только М матрица (3х3).

## 5.4. Содержание отчёта

Отчет по лабораторной работе должен содержать:

- титульный лист;

- задание на лабораторную работу;

- программное и аппаратное окружение при выполнении работы, включая версии Python, библиотеки OpenCV

- тексты программ;

- результаты запуска программ

## 5.5. Контрольные вопросы

1. Перечислите аффинные преобразования.
2. Что такое пирамида изображений?

# Лабораторная работа №6.

# МОРФОЛОГИЧЕСКИЕ ОПЕРАЦИИ

Цель работы: Изучить морфологические операции эрозия, дилатация, размыкание, замыкание, градиент, «цилиндр», «черная шляпа», «успех-неудача»

## 6.1. Общие положения

*Морфологические операции* представляют собой серию операций обработки изображений на основе форм. Они используются для

* предварительной и итоговой обработки изображений (морфологическая фильтрация, утолщение или утончение)
* извлечения свойств изображения (контуров, остовов, выпуклых оболочек)

Эти операции эффективно работают на черно-белых изображениях.

## 6.2. Предварительная подготовка к работе

Ознакомиться с базовыми и производными морфологическими операциями: эрозия, дилатация, размыкание, замыкание

## 6.3. Порядок выполнения работы

Для реализации базовых морфологических операций (эрозия, дилатация) в OpenCV есть специальные функции *erode* и *dilate*, для базовых и производных морфологических операций используется функция *morphologyEx.*

При реализации эрозии используйте следующую функцию

**erode**(src, kernel[, dst[, anchor[, iterations[, borderType[, borderValue]]]]]) -> dst

Параметры:

*src* - входное изображение.

*dst* - выходное изображение

*kernel* - определяет структурный элемент

*anchor* - положение центра внутри структурного элемента. Default = (-1, -1) означает, что центр находится в геометрическом центре.

*iterations* – задает число повторений. Default =1

*borderType -* определяет поведение на краях; если он не задан Default = BORDER\_CONSTANT.

*borderValue* – значение границы в случае постоянной границы

При реализации дилатации используйте следующую функцию

**dilate**(src, kernel[, dst[, anchor[, iterations[, borderType[, borderValue]]]]]) -> dst

Параметры:

*src* - входное изображение.

*dst* - выходное изображение

*kernel* - определяет структурный элемент

*anchor* - положение центра внутри структурного элемента. Default = (-1, -1) означает, что центр находится в геометрическом центре.

*iterations* – задает число повторений. Default =1

*borderType -* определяет поведение на краях; если он не задан Default = BORDER\_CONSTANT.

*borderValue* – значение границы в случае постоянной границы

При реализации базовых и производных морфологических операций используется функция

**morphologyEx**(src, op, kernel[, dst[, anchor[, iterations[, borderType[, borderValue]]]]]) -> dst

Параметры:

*src* - входное изображение.

*dst* - выходное изображение

*op* - тип морфологической операции, который может быть одним из следующих:

MORPH\_ERODE - операция эрозии

MORPH\_DILATE  - операция дилатации

MORPH\_OPEN (2) - операция размыкания

MORPH\_CLOSE (3) - операция замыкания

MORPH\_GRADIENT (4) - морфологический градиент

MORPH\_TOPHAT (5) - «цилиндр»

MORPH\_BLACKHAT (6) - «черная шляпа»

MORPH\_HITMISS - преобразование «успех-неудача»

*kernel* - определяет структурный элемент

*anchor* –положение центра внутри структурного элемента. Default = (-1, -1) означает, что центр находится в геометрическом центре.

*iterations* – задает число повторений эрозии и дилатации. Default =1

*borderType -* определяет поведение на краях; если он не задан Default = BORDER\_CONSTANT.

*borderValue -* значение границы в случае постоянной границы.

При реализации структурных элементов используйте следующую функцию

**getStructuringElement**(shape, ksize[, anchor]) -> retval

Параметры:

*shape* - форма элемента, которая может быть одной из следующих:

MORPH\_RECT (0) - прямоугольный структурный элемент: E\_{i j} =1

MORPH\_CROSS (1) - крестообразный структурный элемент

MORPH\_ELLIPSE (2) - заполненный эллипс, вписанный в прямоугольник Rect (0, 0, ksize.width, ksize.height)

*ksize* - размер элемента. *ksize=* cols, rows – ширина, высота элемента

*anchor* - положение центра внутри структурного элемента. Default = (-1, -1) означает, что центр находится в геометрическом центре.

Структурный элемент необходимо передать в функции *erode (), dilate ()* или *morphologyEx ().*

1. Выполните утончение объекта после однократного применения эрозии на изображении type\_2G.jpg.

Выполните отображение исходного и результирующего изображений.

2. На изображении black\_sq.jpg исключить квадраты с размером стороны меньше 13 пикселей, используя эрозию. Преобразование остальных квадратов не важно.

Выполните отображение исходного и результирующего изображений

3. Выполните утолщение объекта после однократного применения дилатации на изображении type\_2G.jpg.

Выполните отображение исходного и результирующего изображений.

4. На изображении black\_sq.jpg исключить квадраты с размером стороны меньше 13 пикселей, используя дилатацию и эрозию. Остальные элементы неизменны.

Выполните отображение исходного и результирующего изображений.

5. На изображении black\_sq.jpg выполнить дилатацию со следующими структурными элементами

- квадрат,

- крест,

- вида

[0 0 0 1 0]

[0 0 0 1 0]

[1 1 1 1 1]

[0 0 0 1 0]

[0 0 0 1 0]

Все структурные элементы имеют размер (5х5). Используйте различные способы задания структурных элементов

Выполнить отображение исходного и результирующих изображений.

6. Удалите мелкие объекты (удаление шума соль с фона), оставляя крупные на изображении type\_j\_open.jpg, используя операцию размыкания.

Выполните отображение исходного и результирующего изображений.

7. Удалите шум типа перец на изображении type\_j\_close.jpg, используя операцию замыкания.

Выполните отображение исходного и результирующего изображений.

8. Используйте морфологический градиент для сохранения контура края изображения type\_j.jpg.

Выполните отображение исходного и результирующего изображений.

9. Используя операцию TopHat («Цилиндр» ) и различные размеры ядер,

- выделите пятна на изображении type\_j.jpg, которые подсвечиваются больше, чем соседние.

- получите шум изображения type\_j.jpg

Выполните отображение исходного и результирующего изображений

10. Используя операцию BlackHat («Черная шляпа»), выделите области на изображении type\_j.jpg, которые оказываются темнее остальных

## 6.4. Содержание отчёта

Отчет по лабораторной работе должен содержать:

- титульный лист;

- задание на лабораторную работу;

- программное и аппаратное окружение при выполнении работы, включая версии Python, библиотеки OpenCV

- тексты программ;

- результаты запуска программ

## 6.5. Контрольные вопросы

1. К каким последствиям приводит дилатация изображения?
2. К каким последствиям приводит эрозия изображения?
3. Для чего используются морфологические операции?
4. Какая морфологическая операция ликвидирует небольшие отверстия (оставляя более крупные), присутствующие в объекте, сохраняя при этом первоначальный размер объекта неизменным?
5. Какие морфологические операции обработки изображения относятся к базовым?

# Лабораторная работа №7.

# ЧАСТОТНЫЕ ФИЛЬТРЫ

Цель работы: Изучить частотные фильтры для сглаживания изображений (низкочастотные фильтры) и повышения резкости изображений (высокочастотные фильтры)

## 7.1. Общие положения

Методы обработки в частотной области (частотные методы) основаны на модификации сигнала, сформированного путем применения преобразования Фурье к изображению

## 7.2. Предварительная подготовка к работе

Изучить процедуру фильтрации в частотной области:

* умножить *f(x,у)* на *(-1)x+y*
* вычислить прямое ДПФ *F(u,v)*
* умножить F(u,v) на функцию фильтра *H(u,v)*
* вычислить обратное ДПФ
* выделить вещественную часть
* умножить вещественную часть на *(-1)x+y*

## 7.3. Порядок выполнения работы

При реализации фильтрации в частотной области используются следующие функции:

***np.fft.fft2()*** – выполняет двумерное дискретное преобразование Фурье

Используется для вычисления двумерного дискретного преобразования Фурье (DFT) двумерного массива. Он принимает входной массив комплексных чисел и возвращает выходной массив той же формы, содержащий коэффициенты DFT.

В результате получается массив комплексных чисел, где каждый элемент представляет амплитуду и фазу определенной частотной составляющей во входных данных.

***np.fft.fftshift()*** – сдвигает компонент нулевой частоты в центр спектра.

По сути реализует операцию центрирования спектра

***np.fft.ifft2****()* – выполняет обратное двумерное дискретное преобразование Фурье

Возвращает массив комплексных чисел

***np.fft.ifftshift****()* – выполняет действие, обратное *np.fft.fftshift().* Перемещает низкую частоту из центра в верхний левый угол.

***np.linspace()*** - создает последовательность данных, равномерно расположенных на числовой прямой в заданном интервале.

***np.newaxis()* –** позволяет увеличить размерность массива (с 2-х до 3-х, 3-х до 4-х и т.д.)

1. Выполните процедуру фильтрации в частотной области, используя

ФНЧ Баттерворта. Параметры фильтра:n=2, используйте значения *D0* =0.05, 0.5, 10.0

В качестве входного изображения взять файл white\_sq.jpg

Выполните отображение исходного и результирующего изображений, а также амплитудный спектр фильтра

2. На примере использования файла moscow.jpg покажите, что последовательное применение прямого и обратного преобразований Фурье приводят к исходному изображению.

## 7.4. Содержание отчёта

Отчет по лабораторной работе должен содержать:

- титульный лист;

- задание на лабораторную работу;

- программное и аппаратное окружение при выполнении работы, включая версии Python, библиотеки OpenCV

- тексты программ;

- результаты запуска программ

## 7.5. Контрольные вопросы

1. Есть ли взаимно однозначное соответствие между линейными пространственными фильтрами и фильтрами в частотной области?
2. Какой из пространственных фильтров не может быть реализован в частотной области?
3. Что можно сказать о изображении, у которого светлый центр спектра в частотной области

# Лабораторная работа №8.

# ОБНАРУЖЕНИЕ КОНТУРНЫХ ПЕРЕПАДОВ, ВЫДЕЛЕНИЕ КОНТУРОВ, ПОИСК ПРЯМЫХ С ПОМОЩЬЮ ПРЕОБРАЗОВАНИЯ ХАФА

Цель работы: изучить применение базовых функций OpenCV для решения задач поиска краев, границ и контуров.

## 8.1. Общие положения

Резкое изменение яркости изображения представляет интерес по нескольким причинам:

1. Такие изменения, как правило, возникают на границах объектов, например, при изображении светлого объекта на темном фоне или наоборот.

2. Резкие изменения яркости могут быть следствием изменения отражательной способности на достаточно характерных структурах, таких, как разметка пешеходного перехода или пятна на шкуре леопарда.

3. Резкие изменения ориентации поверхности – это еще одна возможная причина изменения яркости.

Точки, в которых происходит резкий перепад яркости изображения, называются краями (ребрами)или краевыми точками.

Когда на границе объекта соединяются все точки, то получается контур. Как правило, конкретный контур относится к граничным пикселям, имеющим одинаковый цвет и интенсивность.

Типичные методы выделения ребер базируются на применении к исходному изображению операторов Собеля и Шарра (операторы первых производных по направлениям) или дискретного оператора Лапласа (оператор вторых производных).

Избирательность выделения контуров можно повысить путем

* сглаживания изображения перед вычислением градиента
* пороговой обработке градиентного изображения

На данный момент наиболее известным детектором ребер является детектор Кэнни.

Операторы градиента, Лапласа и Кэнни позволяет только обозначить на изображении контуры, но не выделить.

Для выделения контуров необходимо повторно обойти изображение, полученное оператором, и на нем найти контуры.

OpenCV позволяет легко находить и рисовать контуры на изображениях. Есть две простые функции:

* findContours()
* drawContours()

## 8.2. Предварительная подготовка к работе

1. Ознакомиться с алгоритмом работы детектора Кэнни.

2. Ознакомиться с преобразованием Хафа для поиска прямых и окружностей

## 8.3. Порядок выполнения работы

1. Применение оператора (детектора) Кэнни

Разработать программу, обеспечивающую загрузку изображения cat.jpg и применение оператора Кэнни**.** В качестве порогов используйте значения 50 и 200.

Выполнить отображение исходного и результирующего изображений.

При реализации используются следующая функция

**Canny**(image, threshold1, threshold2[, edges[, apertureSize[, L2gradient]]]) -> edges

Параметры:

*image* - входное изображение;.

*edges* - выводит изображение того же размера и типа, что и src (выходная карта краев).

*threshold1* - 1ый порог для процедуры гистерезиса.

*threshold2* - 2ой порог для процедуры гистерезиса.

*apertureSize -* размер апертуры для оператора Sobel; default = 3

*L2gradient -* флаг, указывающий, следует ли использовать более точную норму;

default = *false*

*L\_2 norm = sqrt{(dI/dx)2 + (dI/dy)2}* для вычисления величины градиента изображения *(L2gradient = true),* или по умолчанию *L\_1 norm =|dI/dx|+|dI/dy|* достаточно *(L2gradient = false)*

2. Влияние сглаживания изображения перед вычислением градиента

Разработать программу, обеспечивающую следующую последовательность действий для изображений cat.jpg и cat\_gauss\_noise.jpg:

* удаление шумов с помощью фильтра Гаусса
* преобразование полученного изображения в оттенки серого
* применения горизонтального и вертикального операторов Собеля или Шарра (значения весовых коэффициентов указанных градиентов взять по 0.5); или применение фильтра Лапласса или Кэнни (в качестве порогов используйте значения 50 и 200).
* вычисление приближенного значения градиента для двухкомпонентных операторов (Собеля и Шарра)

Выполнить отображение исходного и результирующего изображений для всех операторов.

Сравните с изображениями, полученными без сглаживания (лаб. Работа 4 «Пространственные фильтры»)

При реализации дополнительно используются следующие функции

cv.convertScaleAbs(src) -> dst

вычисляет абсолютное значение и конвертирует его в 8-битовое значение

cv.addWeighted( src1, alpha, src2, beta, gamma) -> dst

вычисляет сумму 2-х массивов с весами по формуле

*dst(I)=saturate(src1(I)∗alpha+src2(I)∗beta+gamma)*

1. Использование функций findContoursиdrawContours

При реализации используйте следующие функции

**findContours**(image, mode, method[,contours[, hierarchy[, offset]]])->contours, hierarchy

Параметры:

*image* - входное 8-битное одноканальное изображение.

*contours* - обнаруженные контуры. Каждый контур хранится как вектор точек.

***hierarchy*** - дополнительный выходной вектор, содержащий информацию о топологии изображения. В нем столько элементов, сколько контуров. Для каждого i-го контура contours [i] элементы

hiearchy [i] [0] - индекс следующего контура на текущем слое Next

hiearchy [i] [1] - индекс предыдущего контура на текущем слое Previous

hiearchy [i] [2] - индекс первого контура на вложенном слое First\_Child

hiearchy [i] [3] - индекс родительского контура Parent

Если для контура i отсутствует Next, Previous, First\_Child, Parent контуры, соответствующий элемент hiearchy [i] будет <0.

***mode*** - режим извлечения контура.

CV\_RETR\_EXTERNAL извлекает только крайние внешние контуры. Он устанавливает hiearchy [i] [2] = hiearchy [i] [3] = - 1 для всех контуров.

CV\_RETR\_LIST извлекает все контуры без установления каких-либо иерархических отношений.

CV\_RETR\_CCOMP группирует контуры в двухуровневую иерархию. На верхнем уровне - внешние контуры объекта. На 2-м уровне — контуры отверстий, если таковые имеются. Все остальные контуры попадают на верхний уровень.

CV\_RETR\_TREE извлекает все контуры и восстанавливает полную иерархию вложенных контуров.

***method*** - метод аппроксимации контура (какие точки внутри контура сохраняются)

CV\_CHAIN\_APPROX\_NONE аппроксимация отсутствует и все контуры хранятся в виде отрезков, состоящих из двух пикселей.

CV\_CHAIN\_APPROX\_SIMPLE склеивает все горизонтальные, вертикальные и диагональные контуры.

CV\_CHAIN\_APPROX\_TC89\_L1, CV\_CHAIN\_APPROX\_TC89\_KCOS применяет одну из разновидностей алгоритма аппроксимации цепочки Teh-Chin.

*offset - н*еобязательное смещение, на которое смещается каждая точка контура.

Функция findContours возвращает список контуров *contours* и иерархию *hierarchy*.

Иерархия контуров представлена в виде массива, каждый элемент которого содержит массив из четырех значений:

 [Next, Previous, First\_Child, Parent]

**drawContours**(image, contours, contourIdx, color[, thickness[, lineType[, hierarchy[, maxLevel[, offset]]]]]) -> image

Параметры:

*image* – выходное изображение.

*contours* – все обнаруженные контуры (контуры, полученные с помощью функции *findContours()*). Каждый контур хранится как вектор точек.

*contourIdx* – параметр, указывающий контур для рисования. Если он отрицательный, рисуются все контуры.

*color* – цвет контуров.

*thickness* – толщина линий, по которым нарисованы контуры. Если он отрицательный (например, thickness = CV\_FILLED), внутренняя часть контура прорисовывается; default =1

*lineType* – связность линии. Подробности смотрите в *LineTypes;* default =8

*hierarchy* – дополнительная информация об hierarchy. Это необходимо только в том случае, если вы хотите нарисовать только часть контуров (see maxLevel).

*maxLevel* – максимальный уровень нарисованных контуров.

Если равен 0, рисуется только указанный контур.

Если равен 1, функция рисует контур (ы) и все вложенные контуры.

Если равен 2, функция рисует контуры, все вложенные контуры, все контуры, вложенные во вложенные, и так далее.

Этот параметр учитывается только при наличии hierarchy.

*offset* – необязательный параметр смещения контура. Сдвинуть все нарисованные контуры на указанное offset=(dx,dy) .

**threshold**(src, thresh, maxval, type[, dst]) -> retval, dst

Применяет порог фиксированного уровня к каждому элементу входного массива

Параметры:

*src -* входной массив*.*

*dst -* выходной массив того же размера и типа, что и *src.*

*thresh -* пороговое значение*.*

*maxval -* максимальное значение для использования с пороговыми значениямиTHRESH\_BINARY и THRESH\_BINARY\_INV*.*

*type -* тип порога

3.1 Найдите наибольший по площади контур на изображении max\_area\_contour.jpg. Его контур выделите синим цветом.

При реализации используйте функцию contourArea

3.2 Найдите наибольший по периметру контур на изображении max\_area\_contour.jpg. Его контур выделите синим цветом.

При реализации используйте функцию arcLength

3.3 Выполните аппроксимацию наибольшего по периметру контура на изображении max\_area\_contour.jpg, используя полигон. Полигон выделите синим цветом

При реализации используйте функции arcLength и approxPolyDP

3.4 Выполните сравнение контуров на изображениях goal\_contour\_star.jpg и max\_area\_contour.jpg

При реализации используйте функцию matchShapes. Максимальную разницу установите равной 1.0.

1. Поиск прямых линий с помощью преобразования Хафа

Разработайте программу, обеспечивающую для изображения image0.jpg

- поиск краев с помощью Кэнни фильтра

- преобразование полученного изображения в оттенки серого

- поиск линий с помощью преобразование Хафа. Используйте 2 варианта с функциями *HoughLines и HoughLinesP.* Установите threshold=100 для HoughLines и threshold=10 для HoughLinesP.

- рисование обнаруженных линий (line)

Выполнить отображение исходного и результирующих изображений для 2-х вариантов.

При реализации используйте функции

**HoughLines**(image, rho, theta, threshold[,lines[,srn[,stn[,min\_theta[,max\_theta]]]]])->lines

Параметры:

*image* - входное изображение;.

*lines* - выходной вектор линий. Каждая строка представлена двухэлементным вектором (ρ, θ). ρ - это расстояние от начала координат (0,0) (верхний левый угол изображения). θ - угол поворота линии в радианах (0 вертикальная линия, pi / 2 горизонтальная линия)

*rho -* дистанционное разрешение (точность) результатав пикселях*.*

*theta -* угловое разрешение (точность) результатав радианах*.*

*threshold -* параметр порога результата. Возвращаются только те линии, которые набрали достаточно голосов (> *threshold*) (минимальную длину строки, которая должна быть обнаружена).

*srn -* для многомасштабного преобразования Хафа. Если и srn = 0, и stn = 0, используется классическое преобразование Хафа. В противном случае оба эти параметра должны быть положительными; default =0

*stn - для многомасштабного преобразования Хафа;* default =0

Альтернативный, более **простой метод прямого извлечения точек**

**HoughLinesP**(image, rho, theta, threshold[, lines[,minLineLength[,maxLineGap]]]) ->lines

*image* - входное изображение;.

*lines* - выходной вектор линий. Каждая строка представлена двухэлементным вектором (ρ, θ). ρ - это расстояние от начала координат (0,0) (верхний левый угол изображения). θ - угол поворота линии в радианах (0 вертикальная линия, pi / 2 горизонтальная линия)

*rho -* дистанционное разрешение (точность) результатав пикселях*.*

*theta -* угловое разрешение (точность) результатав радианах*.*

*threshold -* параметр порога результата. Возвращаются только те линии, которые набрали достаточно голосов (> *threshold*) ) (минимальную длину строки, которая должна быть обнаружена)

*minLineLength -* минимально разрешенная длина строки

*maxLineGap -* максимально допустимый зазор между линиями для их соединения

## 8.4. Содержание отчёта

Отчет по лабораторной работе должен содержать:

- титульный лист;

- задание на лабораторную работу;

- программное и аппаратное окружение при выполнении работы, включая версии Python, библиотеки OpenCV

- тексты программ;

- результаты запуска программ

## 8.5. Контрольные вопросы

1. Назовите методы поиска краев и границ

2. Опишите алгоритм поиска и отображения контуров

3. Сколько пороговых значений используется для определения гистерезисного порога в детекторе Кэнни?

# Лабораторная работа №9.

# СЕГМЕНТАЦИЯ ИЗОБРАЖЕНИЙ

Цель работы: изучить применение пороговой бинаризации и метода выращивания областей для сегментации изображений.

## 9.1. Общие положения

Пороговая бинаризация – это метод обработки изображений, который позволяет преобразовать цветное или полутоновое изображение в черно-белое.

Типы бинаризации: глобальная, адаптивная.

## 9.2. Предварительная подготовка к работе

Ознакомиться с возможными типами бинаризации, изучить простой алгоритм и метод Оцу выбора порога при глобальной бинаризации. Ознакомиться с алгоритмом выращивания областей.

## 9.3. Порядок выполнения работы

1. Глобальная бинаризация изображения

Разработайте программу, обеспечивающую глобальную бинаризацию изображения threshold.jpg. Число на изображении соответствует значению яркости, которым оно изображено.

Выбрать maxval = 255

Выполнить для 2-х типов порога:

-THRESH\_BINARY (значения thresh = 32 и 128)

-THRESH\_OTSU

Выполнить отображение исходного и результирующего изображений

Для реализации используйте функцию

**threshold**(src, thresh, maxval, type[, dst]) -> retval, dst

Применяет порог фиксированного уровня к каждому элементу входного массива

Параметры:

*src -* входной массив*.*

*dst -* выходной массив того же размера и типа, что и *src.*

*thresh -* пороговое значение*.*

*maxval -* максимальное значение для использования с пороговыми значениямиTHRESH\_BINARY и THRESH\_BINARY\_INV*.*

*type -* тип порога. Возможные значения **THRESH\_BINARY**, THRESH\_BINARY\_INV, THRESH\_TRUNC, THRESH\_TOZERO, THRESH\_TOZERO\_INV. Специальные значения THRESH\_OTSU или THRESH\_TRIANGLE могут быть объединены с одним из вышеуказанных значений (cv.THRESH\_BINARY + cv.THRESH\_OTSU). В этих случаях функция определяет оптимальное пороговое значение (параметр *thresh* игнорируется*)*, используя алгоритм Otsu или Triangle, и использует его вместо указанного порога**.**

Возвращаемое значение:

порог *retval -* Если это глобальный алгоритм с фиксированным порогом (например, THRESH\_BINARY), будет возвращено значение параметра threshold.

Если это алгоритм глобального адаптивного порога (например, THRESH\_OTSU), он возвращает соответствующий порог, рассчитанный с помощью адаптивного.

2. Адаптивная пороговая бинаризация изображения

Разработайте программу, обеспечивающую адаптивную пороговую бинаризацию изображения adapt\_threshold.jpg

Проверить на различных размерах blockSize. Выбрать подходящий.

Сравнить с глобальной пороговой бинаризацией Otsu.

Выполнить отображение исходного и результирующего изображений

Для реализации используйте функцию

**adaptiveThreshold**(src, maxValue, adaptiveMethod, thresholdType, blockSize, C[, dst]) -> dst

Параметры:

*src -* входной массив*.*

*dst -* выходной массив того же размера и типа, что и *src.*

*maxval -* максимальное значение для использования с пороговыми значениямиTHRESH\_BINARY и THRESH\_BINARY\_INV*.*

*adaptiveMethod* - адаптивный метод. Возможные значения ADAPTIVE\_THRESH\_MEAN\_C, ADAPTIVE\_THRESH\_GAUSSIAN\_C - среднее и взвешенное среднее значение окрестности *blockSize × blockSize* точки - *C*,

*thresholdType -* тип порога. Возможные значения THRESH\_BINARY, THRESH\_BINARY\_INV

*blockSize* - размер окрестности пикселя, который используется для вычисления порогового значения.

*C -* константа, вычитаемая из среднего или взвешенного среднего

3. Разработать программу для реализации метода выращивания областей на изображении region growing.jpg.

Выращиваемые области: пятиугольник, прямоугольник, 4-х угольная звезда выделить белым цветом.

Использовать 8-связную смежность

**Пояснение**: В качестве центров кристаллизации выбраны точки [(176,255), (229,405), (347,165)] соответственно из пятиугольника, 4-х угольной звезды и прямоугольника

## 9.4. Содержание отчёта

Отчет по лабораторной работе должен содержать:

- титульный лист;

- задание на лабораторную работу;

- программное и аппаратное окружение при выполнении работы, включая версии Python, библиотеки OpenCV

- тексты программ;

- результаты запуска программ

## 9.5. Контрольные вопросы

1. Как рассчитать пороговое значение для адаптивной настройки порога?

2. Какая функция OpenCV используется для адаптивной пороговой обработки?

3. Когда глобальная пороговая обработка дает плохой результат?

4. Назовите основную идею метода выращивания областей.

5. Назовите критерий завершения метода выращивания областей

# Лабораторная работа №10.

# ОПРЕДЕЛЕНИЕ ОБЪЕКТА С ПОМОЩЬЮ ДЕТЕКТОРОВ И КЛАССИФИКАТОРОВ

Цель работы: изучить применение базовых функций OpenCV для решения задач детектирования углов и лиц.

## 10.1. Общие положения

**Особые точки на изображении** — это точки, локальные окрестности которых обладают некоторыми особенностями в сравнении с окрестностями остальных точек изображения. Особые точки каждый алгоритм определяет по-своему.

Процесс определения особых точек достигается путем использования детектора и дескриптора

Один из самых распространённых детекторов особых точек - детектор углов(алгоритм поиска углов на изображении)

Каскадирование основано на объединении нескольких классификаторов, использующих всю информацию, собранную с выхода из одного классификатора, в качестве дополнительной информации для следующего классификатора в каскаде.

Каскадирование является многоступенчатым.

Каскадные классификаторы обучаются с несколькими сотнями «положительных» образцов отдельных объектов и произвольными «негативными» изображениями того же размера. После обучения классификатора его можно применить к области изображения и определить объект, о котором идет речь.

Для поиска объекта во всем кадре окно поиска перемещается по всему изображению с некоторым нахлестом. Этот процесс чаще всего используется при обработке изображений для обнаружения и отслеживания объектов, в первую очередь для обнаружения и распознавания лиц.

Детектирование лиц – получение информации о присутствии лиц и их местоположении на изображении.

Задача детектирования лиц (face detection) на изображении во многих случаях является подзадачей при решении более сложных проблем (например, распознавания лица или определения выражения лица). Информация о присутствии лиц и их местоположении может быть полезна для таких приложений, как охранные системы и системы детектирования лиц в фото- и видеотехнике.

В зависимости от алгоритма решения задачи детектирования лиц решение представляет собой совокупность прямоугольников либо окружностей, окаймляющих лица.

Задача автоматического определения положения лиц является достаточно сложной по следующим причинам.

1. Значительное расхождение внешнего вида и формы лица.

2. Различие изображения лица при изменении ориентации относительно камеры.

3. Вариативность выражения лица.

4. Возможность перекрытия полного изображения, например, прядью волос или полями шляпы.

OpenCV уже содержит много предварительно подготовленных классификаторов для распознавания лица, сохраненных в виде XML файла. Эти файлы XML рас-

положены в папке opencv/sources/data/haarcascades/.

Использование классификаторов рассмотрим на примере детектора лиц и глаз (первый каскад – детектирование лица, второй каскад – детектирование глаз).

Сначала требуется загрузить нужные классификаторы XML и входное изображение (или видео) в монохромном сером режиме.

Затем происходит поиск лица на изображении с использованием загруженных классификаторов. Если лица найдены, то возвращаются их позиции, представленные объектами Rect (x, y, w, h). После получения данных о расположении лиц, создается интересующая область изображения ROI (Region Of Interest) для каждого лица, и поиск глаз происходит только в этих ROI, так как в задаче распознавания лиц учитываются только изображения глаз, расположенные на лицах.

## 10.2. Предварительная подготовка к работе

Ознакомиться с алгоритмом детектора углов Харриса.

Ознакомиться со следующими операциями над изображениями в библиотеке OpenCV: загрузкавидео из файла, извлечение кадров видеопотока, получение и установка свойств видеофайла, сохранение кадров видеопотока в видеофайл, детектирование лиц и глаз на изображении с использованием классификаторов Хаара, отображение прямоугольников и окружностей на изображении

## 10.3. Порядок выполнения работы

1. Детектор углов Харриса

Разработать программу для выделения углов на изображении test\_image\_Smith.jpg, используя детектор углов Харриса. При вызове функции cornerHarris использовать blockSize=2, ksize=3, k=0.04

Оцените, как влияет threshold на количество выводимых точек, выбрав значения threshold 0.01, 0.05, 0.1

При реализации используйте функцию

cornerHarris(src, blockSize, ksize, k[, dst[, borderType]]) -> dst

Параметры:

*src* - входное изображение в оттенках серого и dtype float32;

*dst -* изображение для хранения откликов детектора Харриса

*blockSize -* размер окрестности, учитываемый при определении угла.

*ksize* - параметр апертуры для оператора Sobel().

*k -* свободный параметр детектора Харриса.

*borderType -* метод экстраполяции пикселей

Результатом обнаружения угла Харриса является изображение в градациях серого с оценками интенсивности во всех направлениях.. Пороговое значение соответствующего изображения дает углы изображения

Шаги для обнаружения углов с помощью детектора углов Харриса:

1. Считать входное изображение, используя cv2.imread(). Преобразовать входное изображение в оттенки серого, используя cv2.cvtColor(). Преобразовать тип данных изображения в оттенках серого, используя np.float32.
2. Применить метод cv2.cornerHarris() к изображению в оттенках серого (float32). Передайте методу подходящие blockSize, ksize и k в качестве параметров.
3. Расширить результат для разметки углов и применить пороговое значение для оптимальных значений.
4. Отобразить изображение с обнаруженными углами.

2. Детектирование лица и глаз на изображении с использованием классификаторов Хаара

Разработать программу для детектирования лица и глаз на изображении lena.jpg.

При разработке программы используйте следующую последовательность шагов:

1. Загрузить входное изображение (или видео) в монохромном сером режиме
2. Загрузить натренированные модели лица (классификатор Хаара haarcascade\_frontalface\_alt2.xml или haarcascade\_frontalface\_alt.xml) и глаз (haarcascade\_eye.xml)
3. Выполнить поиск лица на изображении с использованием загруженного классификатора (метод detectMultiScale)
4. Если лица найдены, то вернуть их позиции, представленные объектами Rect (x, y, w, h)
5. Создать интересующую область изображения ROI (Region Of Interest) для каждого лица
6. Выполнить поиск глаз только в этих ROI с использованием загруженного классификатора (метод detectMultiScale).

Функция detectMultiScale принимает изображение в качестве параметра и обнаруживает объекты разных размеров в виде списка прямоугольников.

3. Детектирование лицо и глаза в видеопотоке с использованием классификаторов Хаара

* 1. Выполните базовые операции для видеопотока
* Загрузка видео из файла и извлечение кадров видеопотока.

Открыть видео файл Jalinga\_360.mp4. Для проверки открытия видеопотока используйте метод isOpened. Дополните программу покадровым чтением видеопотока. Определите количество кадров в видеопотоке.

* Сохранение кадров видеопотока в видеофайл.

Дополните программу из предыдущего пункта сохранением кадров видеопотока в новый видеофайл.

3.2 Используйте классификаторы Хаара для видео потока.

Для этого дополните программу из пункта 2 операциями из пункта 3.1 для каждого кадра из видеопотока.

## 10.4. Содержание отчёта

Отчет по лабораторной работе должен содержать:

- титульный лист;

- задание на лабораторную работу;

- программное и аппаратное окружение при выполнении работы, включая версии Python, библиотеки OpenCV

- тексты программ;

- результаты запуска программ;

## 10.5. Контрольные вопросы

1. Назовите способы решения задачи определения объекта.

2. В чем их основные отличия?

3. Что такое особые точки на изображении?

4. Назовите несколько алгоритмов выделения особых точек

# Приложения.

## 1. УСТАНОВКА БИБЛИОТЕКИ OpenCV В СРЕДЕ MICROSOFT VS

Используется следующее окружение:

Операционная система Microsoft Windows 7(10)

Среда разработки Microsoft Visual Studio 2010

Библиотеки OpenCV Версия 2.4.13

Выделяют два основных способа установки библиотеки OpenCV под Windows: посредством установочного файла и из исходных кодов. В данной работе используется первый способ, т.к. он является достаточно простым, интуитивно понятным и не требует значительных усилий.

Чтобы установить библиотеку OpenCV посредством установочного файла, необходимо выполнить следующую последовательность действий:

1. загрузить инсталлятор с официальной страницы библиотеки OpenCV [http://opencv.org/].
2. Запустить установочный файл, выбрать путь для извлечения файлов библиотеки и нажать кнопку «Extract», чтобы инициировать процесс извлечения. По окончании распаковки установочная директория (например, D:/OpenCV2413) будет содержать необходимые заголовочные файлы (директория build/include/opencv2), а также исходные коды библиотеки (sources/modules), набор примеров использования библиотечных функций (sources/samples) и ряд других вложенных директорий.

Остановимся более подробно на составе директории build. Она содержит несколько вложенных директорий, в частности, x86 и x64, в которых находятся сборки библиотеки для соответствующих архитектур. В каждой директории (build/x86 и build/x64) имеются бинарные файлы, собранные под Microsoft Visual Studio (вложенная директория vc14). Файлы с расширением .dll расположены в подпапке bin директория vc14. Файлы с расширением .lib находятся в директории с одноименным названием. Заметим, что бинарные файлы, имеющие приставку 'd' в конце названия, соответствуют Debug-сборке, остальные – сборке в режиме Release. Наряду с этим названия lib-файлов содержат набор цифр, которые определяют версию сборки библиотеки.

1. Задать в переменной окружения PATH пути до dll-файлов библиотеки OpenCV (добавить директорий D:/OpenCV2413/build\x86\vc14\bin): Control Panel → System → Advanced system settings → Advanced Tab → Environment variables... После этого необходимо выполнить restart на компьютере.

Чтобы подготовить среду Microsoft Visual Studio для разработки приложений с использованием OpenCV необходимо выполнить следующую последовательность действий:

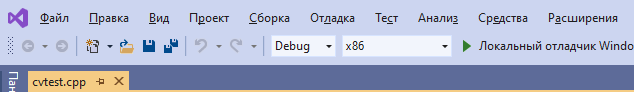
1. Создать новый проект

- запустить приложение Microsoft Visual Studio

- выбрать Файл → Создать → Проект... → Консольное приложение С++, ввести имя проекта (например**,** cvtest) и путь к папке с проектом. Нажать ОК

2. Настройка свойств проекта

- проверить, что "Debug" выбран в окне «Конфигурация»



- Установить пути до заголовочных файлов библиотеки OpenCV

* Проект → Свойства (Properties) → Каталоги VC++ (VC++ Directories)



* выбрать Включаемые директории (Include Directories) и добавить новый D:\OpenCV2413\opencv\build\include
* Нажать ОК для закрытия диалога

- Установить пути до подключаемых библиотек (lib-файлов)

* Вернуться к диалогу Свойства (Properties) → Каталоги VC++ (VC++ Directories). Выбрать Каталоги библиотек (Library Directories) и добавить новый
* D:\OpenCV2413\opencv\build\x86\vc14\lib
* Нажать ОК для закрытия диалога

- Указать список подключаемых программных библиотек

* Вернуться к диалогу Свойства (Properties) → Компановщик (Linker) → Ввод (Input**)** → Дополнительные зависимости (Additional Dependencies) и установить список lib-файлов:

opencv\_calib3d243d.lib

opencv\_contrib243d.lib

opencv\_core243d.lib

opencv\_features2d243d.lib

opencv\_flann243d.lib

opencv\_gpu243d.lib

opencv\_haartraining\_engined.lib

opencv\_highgui243d.lib

opencv\_imgproc243d.lib

opencv\_legacy243d.lib

opencv\_ml243d.lib

opencv\_nonfree243d.lib

opencv\_objdetect243d.lib

opencv\_photo243d.lib

opencv\_stitching243d.lib

opencv\_ts243d.lib

opencv\_video243d.lib

opencv\_videostab243d.lib

Заметьте, что имена файлов с "d" (для "debug").

* Нажать ОК для закрытия диалога

Чтобы использовать функции библиотеки OpenCV при разработке собственных приложений, достаточно подключить заголовочный файл opencv.hpp, содержащий подключение большинства установленных модулей библиотеки, и пространство имен cv, в которое заключены все функции библиотеки.

#include <opencv2\opencv.hpp>

using namespace cv;

Если заранее известно, что в процессе разработки будет использован функционал конкретного набора модулей, то можно подключить только заголовочные файлы соответствующих модулей, например:

#include <opencv2\core.hpp>

#include <opencv2\objdetect.hpp>

using namespace cv;

Чтобы протестировать корректность настроек, необходимо подключить заголовочные файлы библиотеки OpenCV в файл main.cpp как указано выше и скомпилировать программу посредством нажатия клавиши F7.

В процессе компиляции могут возникнуть следующие проблемы:

1. Ошибка открытия заголовочного файла. Примерный текст ошибки приведен ниже. Суть проблемы состоит в том, что указаны неправильные пути до подключаемых заголовочных файлов библиотеки. Решение – проверить в настройках проекта корректность путей до заголовочных файлов

fatal error C1083: Cannot open include file:

'opencv2/opencv.hpp': No such file or directory

2. Ошибка линковки с модулем библиотеки OpenCV. Текст ошибки приведен ниже. Проблема состоит в том, что указаны неправильные пути до подключаемых lib-файлов. Решение – проверить в настройках проекта корректность путей до lib-файлов

LINK : fatal error LNK1104:

cannot open file 'opencv\_core242d.lib'

3. Ошибки линковки, связанные с отсутствием реализации некоторых функций библиотеки OpenCV. Проблема состоит в том, что не подключен модуль (lib-файл), содержащий реализацию функции. Решение – добавить в список lib-файлов необходимый модуль

Когда программа скомпилировалась, запустите ее. Для этого можно нажать сочетание клавиш Ctrl+F5.

## 2. УСТАНОВКА OpenCV-Python НА Windows

Скачайте и установите необходимую версию Miniconda на ПК. "Python + conda + минимальный пакет" <https://conda.io/miniconda.html>. Выберите версию с Python 3.10

Установите последнюю версию pip:

python -m pip install --upgrade pip

Посколько при использовании YOLOv3 совместно с версией OpenCV 4.7.0 возникает ошибка рекомендуется установить версию **4.6.0.66**.

Установите библиотеки Numpy и Matplotlib:

pip install opencv-python==4.6.0.6

pip install matplotlib

Проверьте установку

python

>>> import cv2 as cv

>>> print( cv.\_\_version\_\_ )

4.6.0

>>> exit()

## 3. ИСПОЛЬЗУЕМЫЕ ИЗОБРАЖЕНИЯ

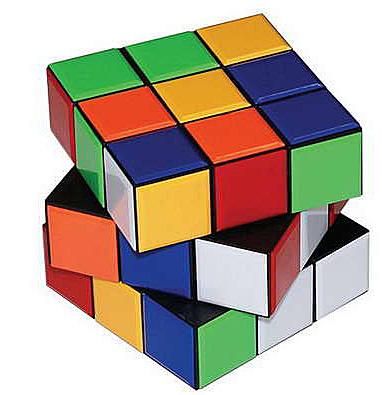


image0.jpg



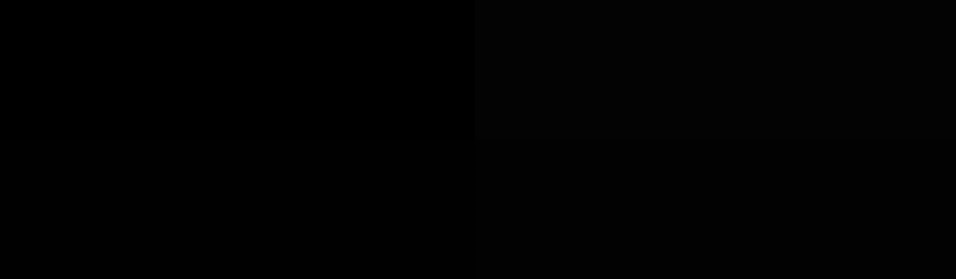
image2.jpg



image3.jpg



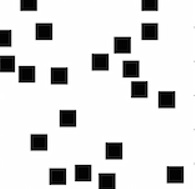
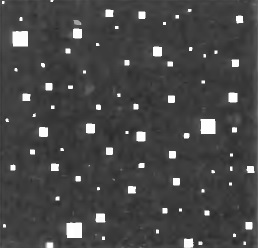
lena.jpg



color4.jpg



planet.jpg

white\_sq.jpg black\_sq.jpg

type\_2G.jpg moscow.jpg



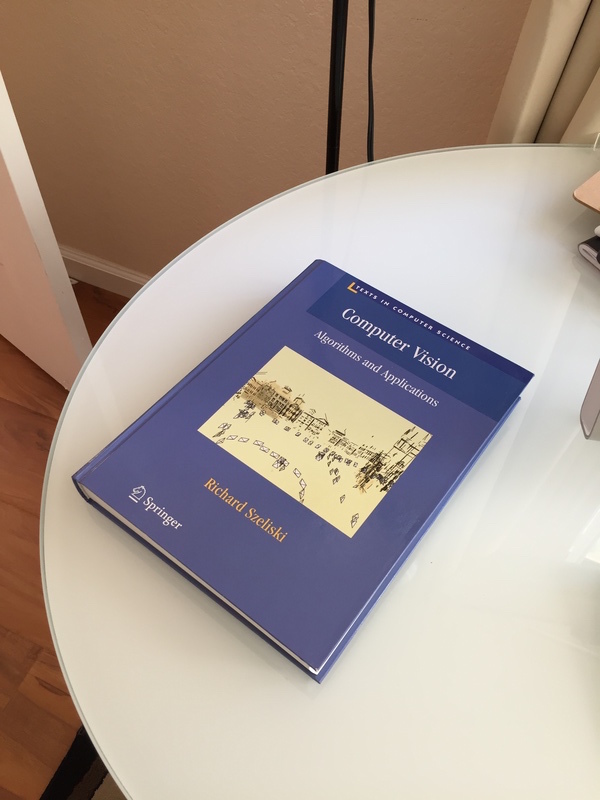
data\_piramid\_up.jpg



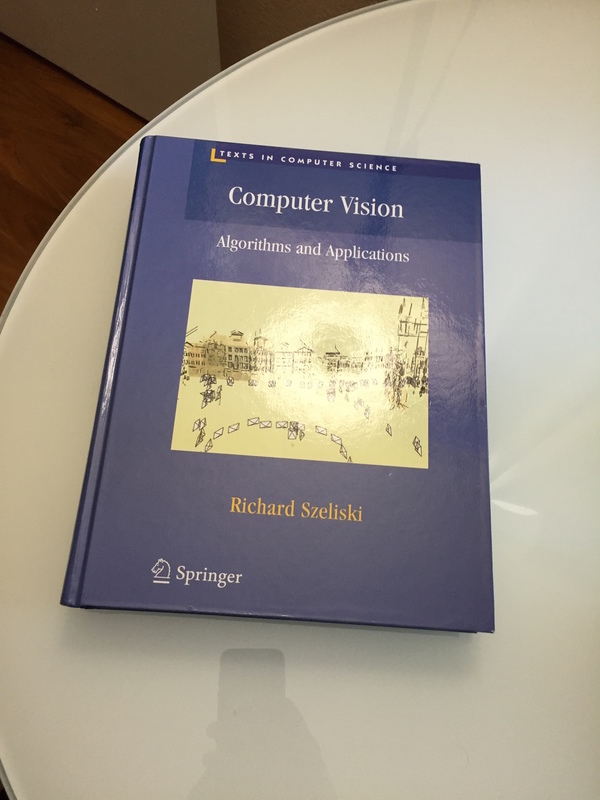
times-square.jpg



first-image.jpg



book1.jpg



book2.jpg

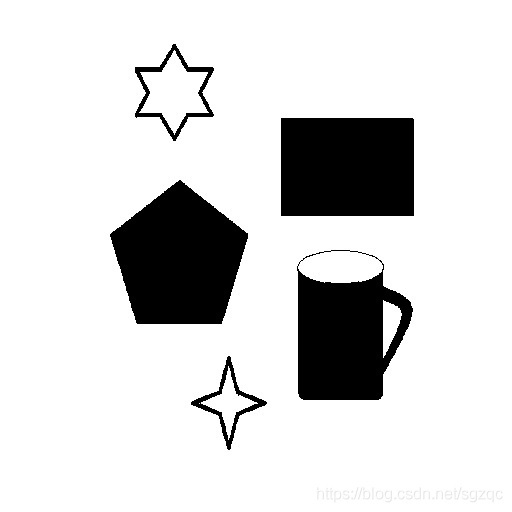
cat.jpg cat\_peppersalt\_noise.jpg



cat\_gauss\_noise.jpg



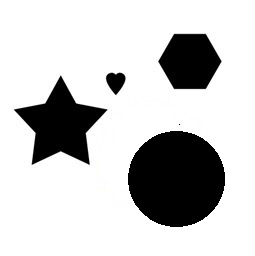
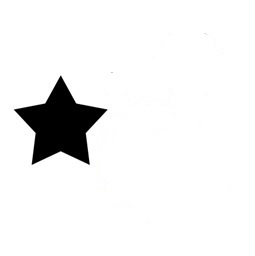
image\_gray\_63-192.jpg



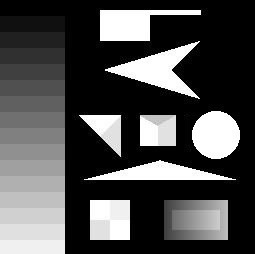
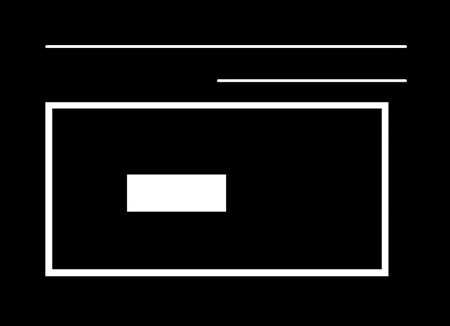
region\_growing.jpg

adapt\_threshold.jpg threshold.jpg

max\_area\_contour.jpg goal\_contour\_star.jpg

test\_image\_Smith.jpg hierarchy.jpg

type\_j\_open.jpg type\_j\_close.jpg type\_j.jpg

Учебно-методическое пособие

Костичев Сергей Валентинович

Электронные методические указания к выполнению лабораторных работ по дисциплине «Обработка изображений и компьютерное зрение (OpenCV)»

Издание публикуется в авторской редакции

СПбГЭТУ «ЛЭТИ»

197376, Санкт*-*Петербург, ул. Проф. Попова, 5