**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра АПУ**

отчет

**по лабораторной работе №1**

**по дисциплине «Программирование систем реального времени»**

|  |  |  |
| --- | --- | --- |
| Студенты гр. 2392 |  | Жук Ф.П. |
| Преподаватель |  | Гульванский В.В. |

Санкт-Петербург

2025

**Цель работы.**

Вам представлен датасет с паттерном, который находится у вас в архиве. Его надо найти на картинке из датасета. Паттерн выглядит так:

Изображение выглядит как символ

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 1 - Патерн

1. Для изображений 1-100 по аналогии с поиском границ необходимо сделать матрицу, которая позволит найти этот паттерн.
2. Для всех остальных изображений необходимо найти любым, рассказанном на лекции, способом этот паттерн. Необходимо описать ваши подходы по нахождению этого паттерна и представить результаты в виде файла, как показано на скриншоте ниже:

**Ход работы.**

Для первых 300 изображений был выбран метод с поиском контуров с помощью библиотеки cv2.

Для поиска центров была написана функция, продемонстрированная на рисунке 1.

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 1 – Функция поиска

Ход работы функции:

1. На вход продаться черно белое изображение (функция чтения cv2.imread(PATH\_IMAGE, cv2.IMREAD\_GRAYSCALE))
2. Далее проводиться бинаризация изображения (пример работы функции продемонстрирован на рисунке 2).

Изображение выглядит как текст, снимок экрана, Прямоугольник, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 2 - пример работы cv2.threshold с параметром cv2.THRESH\_BINARY

1. Далее был выполнен поиск контуров на бинаризованом изображении. Он был выполнен с параметрами .

Функция извлекает контуры из бинарного изображения с помощью алгоритма “ Topological Structural Analysis of Digitized Binary

Images by Border Following”. Алгоритм сканирует изображение построчно в поисках граничного пикселя. Пиксель является граничным, если среди его 4-соседей есть хотя бы один фоновый пиксель. При обнаружении пикселя, принадлежащего новой границе, запускается процедура трассировки. Алгоритм обходит границу по часовой стрелке для внешних границ объектов или против часовой стрелки для границ отверстий внутри объектов, запоминая координаты пикселей границы. Трассировка продолжается, пока алгоритм не вернется в стартовый пиксель.

Параметры:

* cv2.RETR\_EXTERNAL – выделяет вешний контур;
* cv2.CHAIN\_APPROX\_SIMPLE - упрощает контур.

1. Далее рассчитывается момет изображения. Моменты изображения в компьютерном зрении — определённые средневзвешенные значения интенсивности пикселей изображения, или функция таких моментов. Для всех точек контура вычисляются моменты порядка (p+q). Для контура — суммирование по всем его точкам.

Где в итоге m00 - площадь объекта, m10 - суммы x, m00 - суммы y.

1. Зная площадь объекта, суммы x, суммы y рассчитываем центр объекта.

Пример работы кода (рис. 3).

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как текст, снимок экрана, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как снимок экрана, текст, символ, логотип

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как снимок экрана, текст, символ, графический дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 3 - Пример работы

Для изображений 300–400 был выбран симбиоз алгоритмов.

Первый алгоритм – это свертка изображения с последующим нормированием (рис. 4), для фильтрации изображений.

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 4 – Алгоритм свертки

Пример сверток продемонстрирован на рисунке 5.

Изображение выглядит как собака, млекопитающее, черно-белый

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, графический дизайн, черный

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 5 - Пример сверток

Для работы был выбран фильтр черного, он оставляет только пиксели, которые имею яркость 0. Данный фильтр упростит дальнейшую детекцию.

В качестве второго алгоритма выступает cv2.matchTemplate (сегмент кода рис. 6). Это метод поиска и нахождения местоположения шаблонного изображения на более крупном изображении. Она просто скользит по шаблонному изображению и сравнивает шаблон и фрагмент входного изображения под шаблонным изображением. В коде иметься функция масштабирования шаблона.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 6 – Фрагмент кода

Третий алгоритм это SIFT. Это алгоритм для поиска и описания локальных особенностей на изображениях. Его основная задача — находить характерные точки, которые устойчивы к масштабированию, повороту и частично к изменению освещения.

Он работает по следующему алгоритму:

1. Поиск экстремумов в пространстве масштабов. Изображение размазывается с разными масштабами, ищутся точки, которые являются максимумом или минимумом среди соседей.
2. Определение точного положения ключевой точки. Для каждой найденной точки уточняется положение и отбрасываются слабые и неустойчивые точки, например: на краях.
3. Определение ориентации. Для каждой точки вычисляется направление градиента, чтобы сделать описание инвариантным к повороту.
4. Построение дескриптора. Вокруг каждой точки строится вектор признаков, который описывает локальную структуру изображения.

Для поиска совпадающих дескрипторов между пятерном и изображением, используется алгоритм BFMatcher.  Это простой способ сопоставления дескрипторов между двумя изображениями. Он сравнивает каждый дескриптор первого изображения со всеми дескрипторами второго и находит наиболее похожие пары.

Дальше проходит отсеивание ложных совпадения и оставить только действительно похожие участки. Если таких участков меньше 4, то этот участок отсеивается.

В итоге мы получаем центр нашего изображения.

Пример работы продемонстрирован на рисунке 7.

Изображение выглядит как млекопитающее, собака, домашнее животное, Порода собаки

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как млекопитающее, Порода собаки, собака, домашнее животное

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как снимок экрана, белый, черный, текст

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как млекопитающее, собака, волк, собачьи

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как зарисовка, снег, рисунок, черно-белый

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 7 – Результат работы кода

**Выводы.**

В ходе лабораторной работы было проведено исследование основных алгоритмов компьютерного зрения. Были определены центры паттернов на 400 изображениях. В ходе работы применялись корреляция с шаблоном, методы сопоставления ключевых точек SIFT, предобработка изображения различными фильтрами и созданной матрицей для выделения паттерна.

В процессе экспериментов были протестированы различные ядра свёртки для выделения характерных особенностей изображений, а также реализованы методы бинаризации и поиска контуров для автоматического определения центров объектов. Для повышения точности поиска паттернов использовались масштабирование шаблона и фильтрация ложных совпадений с помощью правила Лоу при сопоставлении дескрипторов SIFT.

Результаты работы были сохранены в виде таблицы с координатами найденных центров, что позволяет использовать полученные данные для дальнейшего анализа или обучения моделей машинного обучения. Проведённая работа показала эффективность комплексного подхода, сочетающего классические методы обработки изображений и современные алгоритмы поиска ключевых точек.