Московский государственный университет
имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики

Отчет по лабораторной работе №3

«Изучение и освоение методов обработки и сегментации изображений»

По курсу «Обработка и распознавание изображений»

Выполнил:
студент 317 группы
Дьяков И. А.

Москва
2024

Содержание

1 Постановка задачи

Разработать и реализовать программу для классификации изображений моделей графов, построенных из магнитной головоломки, обеспечивающую:

- Ввод и отображение на экране изображений в формате JPEG;
- Сегментацию изображений на основе точечных и пространственных преобразований;
- Генерацию признаковых описаний структуры графов на изображениях;
- Построение классификатора изображения в соответствии с заданным набором эталонов.

Задача состоит в построении меры сходства изображений на основе выделения и анализа формы графа, составленного из деталей головоломки. Нужно разработать и реализовать алгоритм, входом которого является изображение, а выходом – описание признаков формы изображенной модели. Полное решение предполагает разработку алгоритма, который сможет классифицировать тестовые картинки по всем 4 типам графа.

В качестве признакового описания формы предлагается построить описание топологической структуры графа в виде вектора, в котором k-я компонента есть число вершин степени k в представленном графе. Например, для первого изображения вектор имеет следующий вид: (3, 4, 3, 3), т.е. 3 вершины степени 1 (терминальные вершины), 4 вершины степени 2, 3 вершины степени 3 и 3 вершины степени 4. Здесь нетерминальным вершинам соответствуют шарики, в которых сходятся рёбра графа.

2 Описание данных

В качестве исходных данных прилагается набор из 16 цветных изображений моделей, построенных из деталей магнитной игры-головоломки в формате 1024×768 с разрешением 72 dpi. Всего задано 4 структуры графа, эталоны которых представлены на изображениях 2.jpg (класс I), 3.jpg (класс II), 4.jpg (класс III), 5.jpg (класс IV). На остальных изображениях представлены графы, изоморфные четырём эталонным образцам. Примеры входных изображений представлены на рисунке.

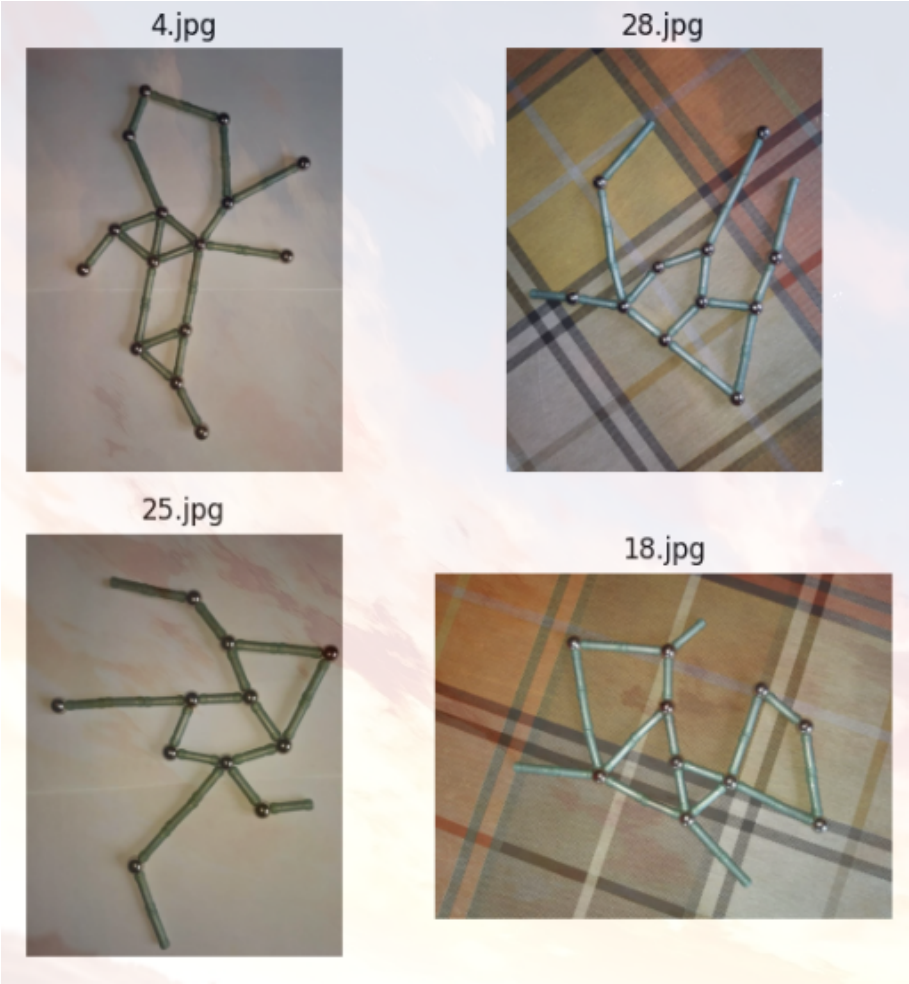


Рис. 1: Пример изображений из предоставленных данных

В задание входят задачи двух уровней сложности: Intermediate, Expert.

- Класс Intermediate: Решение задачи для изображений на белом фоне.
- Класс Expert: Решение задачи для изображений на пёстром фоне.

3 Описание метода решения

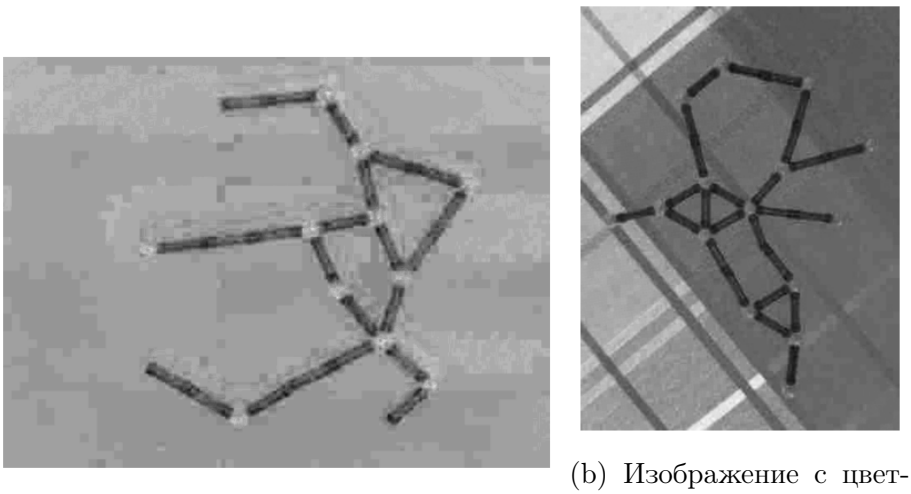
Решение задачи было разбито на следующие основные этапы:

1. Загрузка изображения
2. Бинаризация изображения
3. Построение скелета бинаризованного изображения
4. Выделение ребер графа
5. Выделение вершин графа
6. Построение матрицы смежности
7. Выделение признаков графа из матрицы смежности

3.1 Бинаризация изображения и построение скелета

Основной идеей всего последующего алгоритма является выделение именно ребер графа в отдельности. Данный подход позволяет с хорошей точностью искать вершины графа (как точки пересечения ребер), а также позволяет декомпозировать граф и построить его однозначное представление с точностью до изоморфизма в виде матрицы смежности.

Для выделения ребер графа использовалось специальное цветовое представление изображения, CIELAB. Точнее, использовался второй слой этого представления, на котором, как видно на рисунке, хорошо контрастируют именно ребра графа.



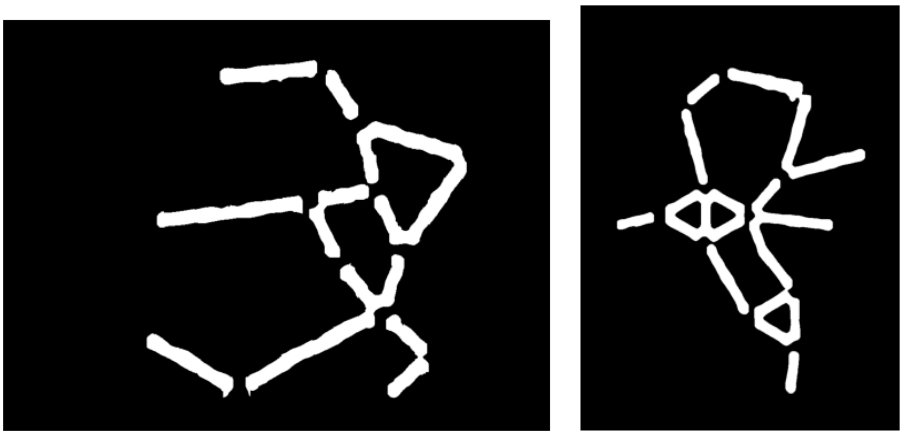
(a) Изображение с монотонным фоном

(b) Изображение с цветным фоном

Рис. 2: Пример представления изображений после преобразования цветовых каналов

Было замечено (см. **??**), что для изображений с пестрым фоном и изображений с монотонным фоном лучше подходят различные подходы бинаризации. Для решения данной проблемы, изображение также переводится в цветовой формат HSV и далее вычисляется дисперсия второго канала изображения (Saturation). Эксперименты показали, что по значению дисперсии сатурации изображения можно безошибочно разделить всю выборку на изображения с монотонным и пестрым фоном.

Для изображений с пёстрым фоном используется подход с глобальной бинаризацией с фиксированным порогом. Для изображений с монотонным фоном используется дополнительное размытие фона и бинаризация методом Оцу. Данные подходы показали наилучшие результаты на предоставленных данных.



(a) Бинаризация изображения с монотонным фоном

(b) Бинаризация изображения с цветным фоном

Рис. 3: Пример представления изображений после преобразования цветовых каналов

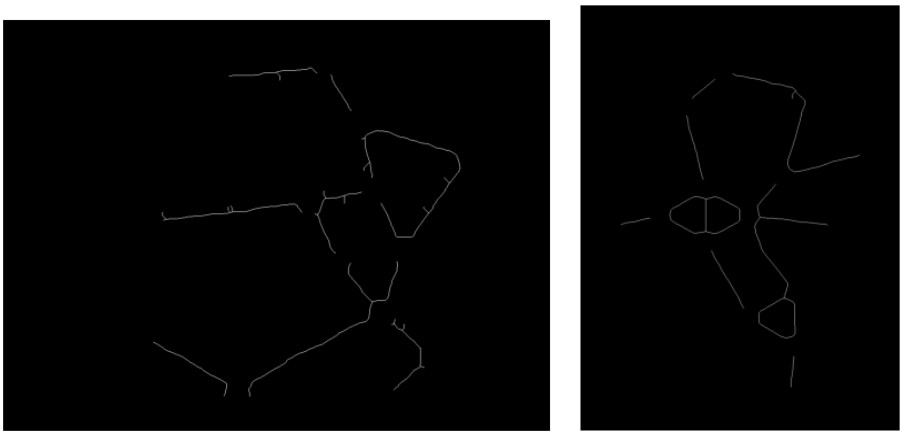
Перед бинаризацией изображение сглаживается комбинацией медианного фильтра и фильтра Гаусса в следующей последовательности:

1. фильтр Гаусса с ядром (9, 9)
2. медианный фильтр с $k = 15$

После бинаризации производилась постобработка, основной целью которой было облегчение последующего выделения скелета. Для этого сначала применялась дилатация для заполнения пустот внутри бинаризованных ребер графа. Затем несколько раз применялась морфологическая операция открытия для удаления шумов и сглаживания ребер.

Все дальнейшие действия уже не разделялись для цветного и монотонного фона, так как эта информация терялась на этапе бинаризации. Далее из

бинаризованного изображения выделяется скелет. Для выделения скелета используется функция `skeletonize` из библиотеки `scikit-image`. При выделении скелета используется метод Zhang для изображений с одним цветовым каналом.



(a) Скелет изображения с монотонным фоном

(b) Скелет изображения с цветным фоном

Рис. 4: Пример скелетонизации изображений

3.2 Выделение ребер графа

Для выделения ребер графа предлагается следующий итерационный процесс:

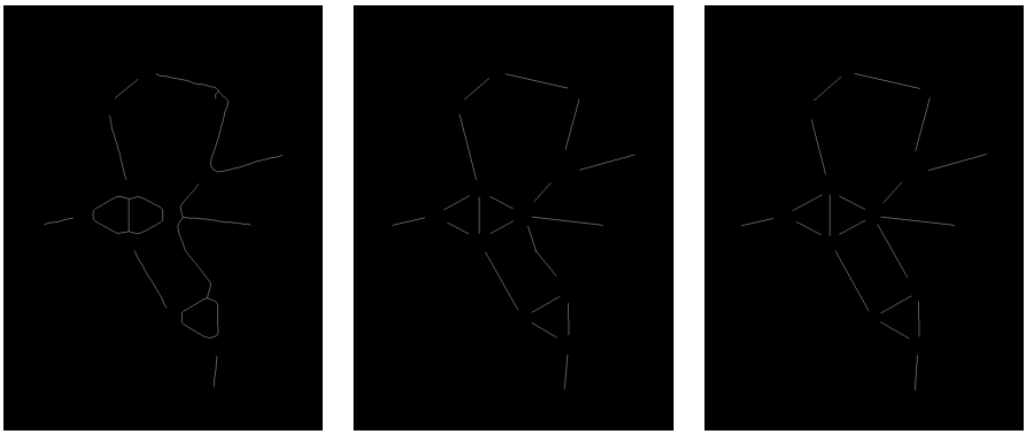
1. Получение прямых линий с помощью преобразования Хафа
2. Поиск, объединение и сглаживание линий, лежащих на одном ребре

На этапе преобразования Хафа в скелете изображения ведется поиск прямых, являющихся ребрами графа. Так как изображение скелета может быть сильно зашумлено (как, например, на рис. 4b), поэтому ребра могут быть изогнутыми и иметь шумовые «отростки». Поэтому преобразование Хафа производится с низким порогом на качество прямой, так как на данном этапе важнее полнота результата, нежели его точность.

Следующий этап исправляет и фильтрует некорректные прямые, которые были выделены на первом шаге. Также, на данном этапе производится объединение всех близких в смысле угла наклона и наличия точек касания прямых в одну прямую. Одной из важных особенностей данного шага является то, что он искусственно удлиняет ребра в обоих направлениях. Это сделано для того, чтобы во время преобразования Хафа не пропадали ребра, так как данное преобразование имеет особенность в укорачивании выделяемых прямых.

Условия останова для данного итерационного алгоритма могут быть разными. Одним из условий может быть то, что алгоритм сошелся, то есть

$$\exists k_0, \forall k > k_0 : \quad ||A^{k+1} - A^k|| < \varepsilon$$



(a) Скелет изображения (b) Представление графа после первой итерации (c) Представление графа после второй итерации

Рис. 5: Пример скелетонизации изображений

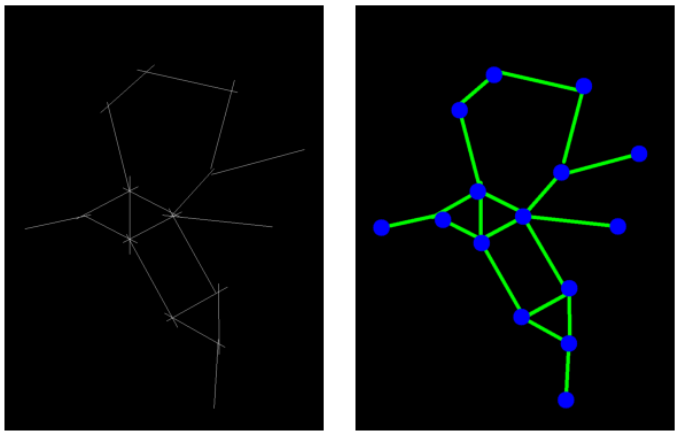
для заранее заданного ε. Однако на практике, при плохом подборе величины «удлинения» ребер, алгоритм часто сходится к результату, в котором отсутствуют некоторые ребра графа (слишком низкий показатель удлинения), или изображение графа сильно зашумлено (слишком высокий показатель удлинения). Поэтому в авторском алгоритме используется двухшаговая модификация данного алгоритма.

3.3 Выделение вершин графа

Для выделения вершин в получившемся изображении полученные на предыдущем этапе обработки ребра дополнительно удлинялись с таким коэффициентом, чтобы ребра, входящие в одну вершину, начали пересекаться. После этого наивным перебором ищутся все точки пересечения ребер и составляются пары пересекшихся ребер (все ребра случайно занумеровываются от 1 до M, где M — количество ребер). Также в качестве вершин добавляются концы всех ребер, для нахождения листовых вершин.

Далее все найденные точки кластеризуются, удаляются все точки, находящиеся слишком близко (совпадающие вершины), и получившиеся в результате точки объявляются вершинами графа. Точки, совпадающие с объявленными вершинами, объединяются (т.е. все ребра, которые проходили через эти точки, присваиваются новой вершине).

Отметим, что данный этап алгоритма можно реализовать эффективнее, так как сейчас поиск всех точек пересечения происходит за $O(M^2)$, в то время как поиск пары пересекающихся отрезков алгоритмом заметающей прямой работает с асимптотикой $O(M \log(M))$. Однако в условиях данной задачи, где число M невелико, ускорение данного этапа алгоритма не дало бы существенного улучшения производительности.



(a) Граф с удлиненными ребрами

(b) Сегментированный граф

Рис. 6: Пример поиска вершин графа и построение итогового представления графа

3.4 Выделение признаков и классификация графа

После нахождения вершин графа достаточно пройтись по всем вершинам и проверить общие ребра для каждой пары вершин. В результате получится описание графа с точностью до изоморфизма, представленное в виде матрицы смежности. Используя матрицу смежности, мы имеем большой выбор признаковых описаний, начиная с самой матрицы, заканчивая сжатыми описаниями с использованием эмбеддингов.

Задача классификации графа решалась с использованием признаковых описаний, предложенных в постановке задачи. Получившиеся после обработки изображения признаки (заметим, что эти признаки легко получить из матрицы смежности графа, например, операцией `np.bincount`) сравнивались с эталонными признаками каждого из четырех типов графов. На основании нормы разности векторов эталонных признаков и полученных признаков выбирался наиболее близкое признаковое описание и класс эталона присваивался классу текущего изображения. Разметка для эталонов производилась вручную.

4 Описание программной реализации

Программа реализована на языке `Python`. Графический интерфейс написан с использованием фреймворка `tkinter`. Программа написана для платформы Linux. Все решение обернуто в docker-контейнер для упрощения работы с зависимостями. Для тестирования реализации написана проверка на сборку контейнера.

Инструкция для запуска, а также весь код проекта находится на странице репозитория проекта `https://github.com/revit3d/Graph-Classification-opencv/tree/trunk`.

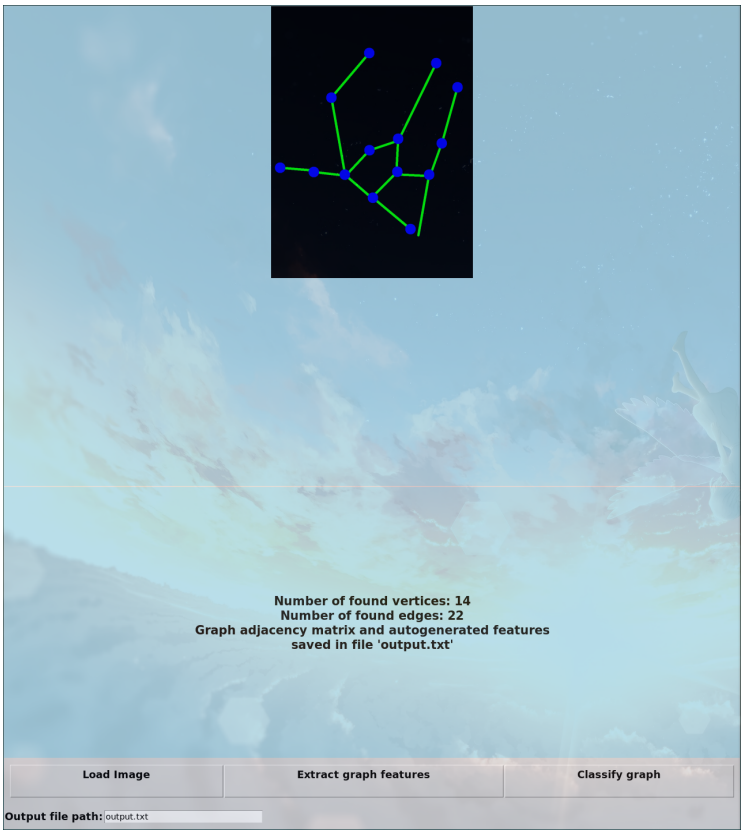


Рис. 7: Интерфейс приложения

Программа предоставляет возможность выбора и загрузки изображения из папки `images`, а также запуск обработки изображения. Результаты обработки записываются по указанному пользователем пути.

Все точечные и пространственные преобразования изображений проводились с использованием библиотеки `OpenCV`. Для получения скелета изображения использовалась библиотека `scikit-learn`. В качестве вспомогательной библиотеки для изменения размера изображения и его демонстрации в графическом интерфейсе использовался python-пакет `Pillow`.

5 Эксперименты

Эксперименты проводились как с целью улучшения качества сегментации графа, так и с целью улучшения качества поиска вершин. Эксперименты включали в себя как комбинирование и тестирование новых алгоритмов сегментации, так и подбор гиперпараметров для улучшения качества работы программы. Ниже приводится список из некоторых проведенных экспериментов.

- Сравнительный анализ различных цветовых схем для улучшения качества сегментации.
- Исследование алгоритмов бинаризации и их комбинаций для сегментации объектов на изображении.
- Исследование зависимости качества сегментации от использования различных фильтров сглаживания и их комбинаций, подбор эффективных параметров сглаживания.
- Подбор условий для разделения изображений с монотонным фоном и изображений с пестрым фоном.
  Данный эксперимент включал в себя исследование различных цветовых схем и подходов к выделению фона. Исследование HSV формата изображения показало, что по всем трем каналам выборки с монотонным и пестрым фоном линейно разделимы по величине среднеквадратичного отклонения величин в каждом из каналов. Это позволило автоматически определять, какой из подходов к бинаризации лучше использовать с тем или иным изображением.
- Эксперименты с удлинением/укорачиванием прямых для их стабилизации при использовании преобразования Хафа.
- Подбор эффективной постобработки для удаления шумов из бинаризованного изображения.

Все эксперименты сохранены в файле `Experiments.ipynb`.

6 Выводы

В процессе экспериментов были исследованы различные подходы к сегментации и выделению контуров на изображении, исследованы возможности скелета изображения для изучения структуры объектов. Было проанализировано влияние точечных и пространственных преобразований и их композиций на изображение. Были разработаны и протестированы алгоритмы сегментации и классификации, использующие свойства исследуемых данных. Выполненная работа и проведенные эксперименты демонстрируют, что использование специфических для конкретной задачи подходов может быть эффективнее и качественнее по сравнению с использованием общих практик.

Данный проект демонстрирует, что эффективные и интерпретируемые преобразования над пикселями изображений могут успешно справляться с задачей сегментации и демонстрировать качество работы, сравнимое со сложными моделями сверточных нейронных сетей и другими значительно менее производительными и сложными подходами к решению данной задачи. Кроме того, данные алгоритмы способны работать без предварительной разметки асессорами, что делает их более выгодными с финансовой точки зрения, а также решает проблемы, связанные с некачественной разметкой данных.