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

Отчет по лабораторной работе №2

«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 работы с изображениями фишек игрового набора Триомино, обеспечивающую:

- Ввод и отображение на экране изображений;
- Сегментацию изображений на основе точечных и пространственных преобразований;
- Поиск фишек на изображениях;
- Классификацию фишек на изображениях.

Входом программы является изображение в формате BMP24. Выходом программы является текстовый файл, в котором каждая запись описывает положение и код одной фишки в следующем формате: N - количество фишек на картинке, X, Y; m1, m2, m3; Где (X, Y) - координаты центра фишки на изображении (X - номер столбца, Y - номер строки), m1, m2, m3 - код фишки — количество точек в углах треугольника.

2 Описание данных

Данные представляют собой изображения различного размера, на которых изображены фишки для игры в Триомино. Фишки могут быть расположены как на однородном, так и на неоднородном фоне, а также может присутствовать неоднородное освещение. Пример входных данных представлен ниже.

(a) Изображение 1 (b) Изображение 2

Рис. 1: Пример изображений из предоставленных данных

3 Описание метода решения

Решение задачи было разбито на следующие основные этапы:

1. Загрузка изображения
2. Поиск фишек Триомино на изображении
3. Выделение отдельных изображений каждой найденной фишки
4. Классификация фишки по ее изображению
5. Агрегация и запись в файл результатов

3.1 Поиск фишек Триомино на изображении

Поиск фишек осуществлялся с помощью следующего алгоритма:

Рис. 2: Исходное изображение

1. **Сглаживание изображения**
   Изображение сглаживалось комбинацией фильтра Гаусса и медианного фильтра. Данный подход позволял эффективно убрать шумы, оставляя больше информации на изображении по сравнению с использованием перечисленных фильтров по отдельности.

2. **Получение контуров с помощью адаптивной бинаризации**
   Адаптивная бинаризация показывает хорошие результаты для изображений с неравномерным освещением. После бинаризации производилась постобработка с помощью фильтра открытия. Из бинаризованного изображения выделялись контуры.

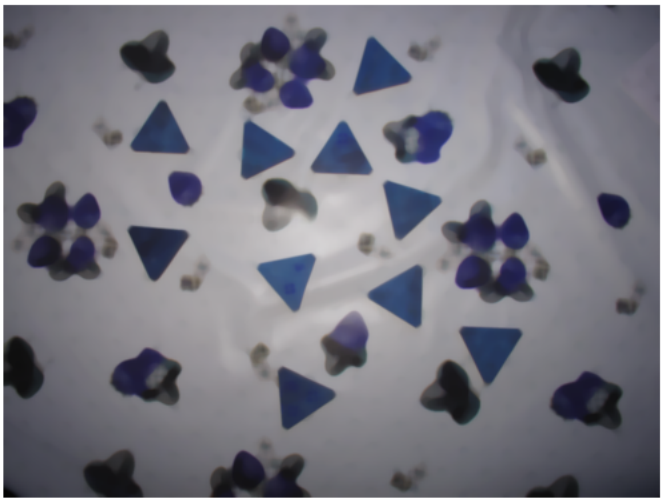

Рис. 3: Сглаживание изображения

3. **Получение контуров с помощью бинаризации методом Отцу**
   Данный подход к бинаризации позволял выделить некоторые контуры, не выделяемые с помощью адаптивной бинаризации. Проблема использования каждого из подходов по отдельности заключается в том, что при решении данной задачи необходимо соблюдать баланс удаления шумов и сохранения полезной информации в изображении. Комбинация данных подходов позволила исправлять ошибки каждого из алгоритмов и, таким образом, добиться лучшего качества сегментации. После бинаризации производилась постобработка с помощью фильтра открытия. Из бинаризованного изображения выделялись контуры.

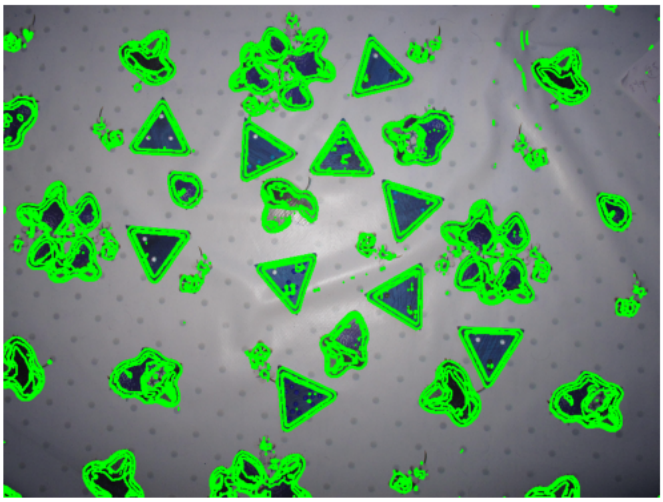

Рис. 4: Выделение контуров комбинацией методов

4. **Постобработка контуров**
   Сегментированные на предыдущих двух шагах контуры объединялись в

одно множество и приближались многоугольниками. Далее отбрасывались те контуры, для которых наилучшим приближением не являлись треугольники, близкие к равносторонним в смысле соотношения длин сторон. Также, отбрасывались накладывающиеся треугольники (так как во многих случаях оба алгоритма находили одну и ту же фишку).

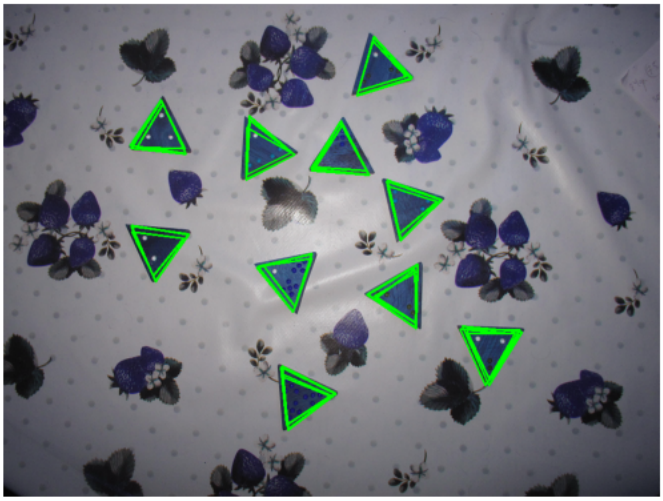

Рис. 5: Аппроксимация и фильтрация контуров

В результате работы данного алгоритма получались треугольные области, являющиеся контурами сегментированных фишек.

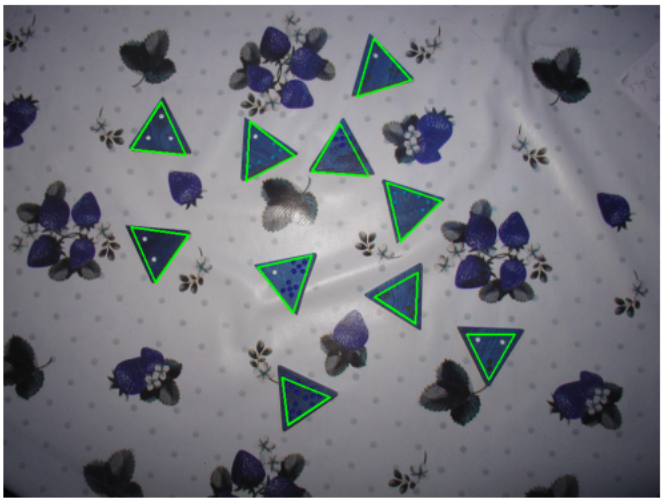

Рис. 6: Удаление совпадающих контуров

Важным улучшением алгоритма по сравнению с переводом изображения в черно-белый формат стала обработка по отдельности по каждому из каналов RGB и объединение результатов бинаризации с помощью попиксельного применения операции побитового «или». В результате возросла стабильность и качество работы алгоритма.

3.2 Поиск фишек на изображении

Перед непосредственной классификацией сегментированных фишек изображение каждой фишки вырезалось из исходного и поворачивалось так, чтобы произвольно выбранная сторона треугольника находилась в нижней части изображения параллельно оси OX изображения, а сам треугольник занимал все изображение (то есть ширина изображения совпадала с длиной стороны треугольника, а высота — с высотой треугольника).

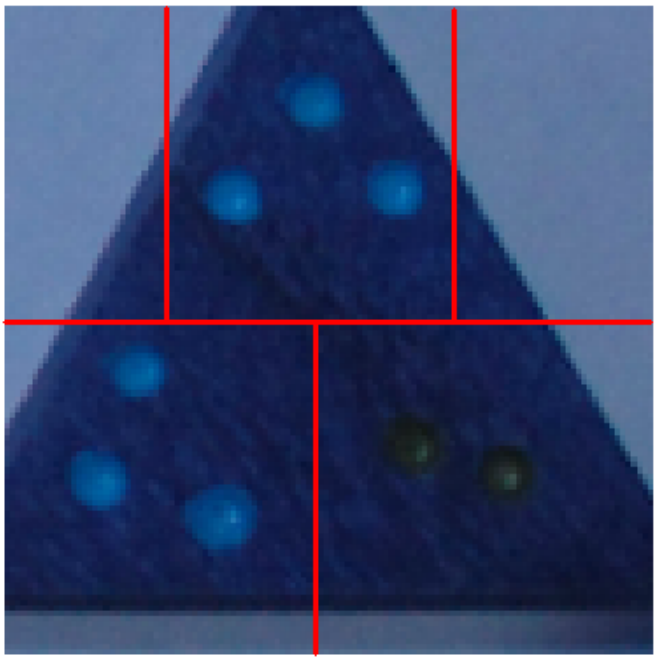

Рис. 7: Пример вырезанного изображения отдельной фишки
Линиями выделены контуры обрезания для каждой вершины

Выполнение данных инвариантов для изображения фишки позволяет утверждать, что:

- Мы можем однозначно разделить изображение на секторы, в которых будут находиться отдельные маркировки каждой из вершин фишки.
- Все изображения фишек имеют приблизительно одинаковое разрешение (для одного изображения)
- На изображении не будет посторонних объектов, которые могут ухудшить качество сегментации.

Таким образом, данное преобразование является ключевым для решения задачи классификации фишки Триомино. После вырезания и поворота изображений, каждое из них обрабатывалось независимо. Отметим, что это позволяет эффективно решать задачу с применением многопоточности.

Каждое изображение разрезалось на три (вершины фишки), и затем также обрабатывалось независимо. Дальнейший алгоритм выделения маркировки представлен ниже:

1. **Сглаживание изображения**
   Изображение сглаживалось с помощью медианного фильтра. Так как рассматриваемые изображения имеют низкое разрешение, чрезмерное сглаживание убирает полезную информацию из изображения, делая дальнейшую сегментацию невозможной. Поэтому сглаживание производилось с минимальным эффективным фильтром размера 3.

2. **Адаптивное осветление изображения**
   Усреднялось значение пикселей по всем каналам и на основании этой информации изображение осветлялось или затемнялось на коэффициент, зависящий от того, насколько обрабатываемое изображение засвечено или затемнено.

3. **Выделение информативных признаков из изображения** Входное изображение в формате RGB преобразовывалось в цветовой формат YCBCR. Далее объединялись каналы исходного изображения и преобразованного (6 каналов), и к ним добавлялись негативы каждого из каналов (12 каналов). Из данных каналов отбрасывались шумовые, остальные каналы использовались для независимого поиска контуров, и результаты объединялись с помощью попиксельного применения операции побитового «или».

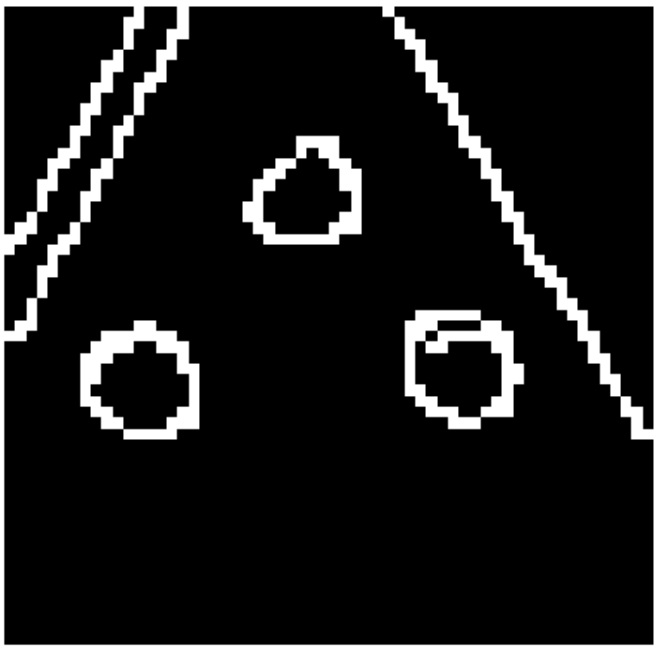

Рис. 8: Результат поиска контуров для отдельно взятой вершины фишки

4. **Поиск контуров**
   Поиск контуров осуществлялся при помощи алгоритма Кенни. Были протестированы и другие подходы, в том числе те, которые были использованы при сегментации фишек. Однако другие подходы показали более высокую зависимость от шума на изображении с низким разрешением, поэтому от них пришлось отказаться.

5. **Приближение контуров окружностями и фильтрация** Полученные контуры приближались окружностями и далее фильтровались на основании радиуса лучшего приближения (фильтрация шумов и контуров самой фишки) и позиции центра окружности (фильтрация совпадающих точек).

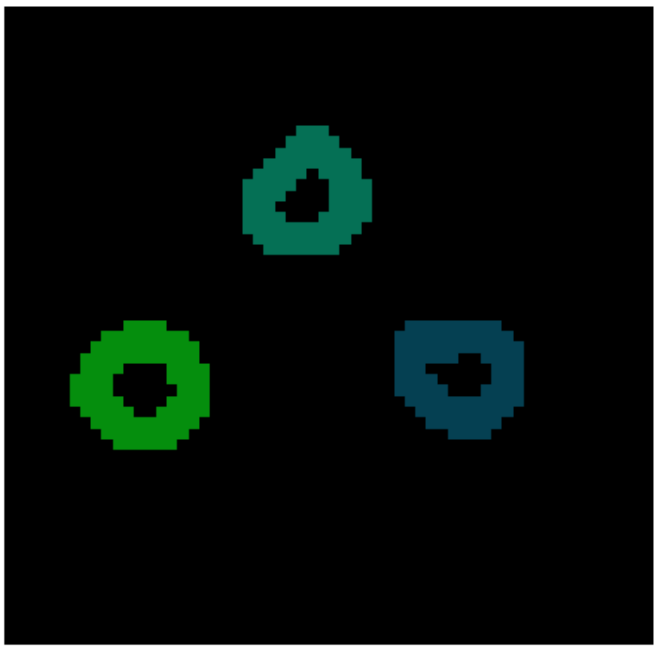

Рис. 9: Результат фильтрации контуров

Полученные окружности подсчитывались и возвращались в качестве маркировки обработанной вершины.

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/Triomino-Classification-opencv/`

`tree/trunk`.

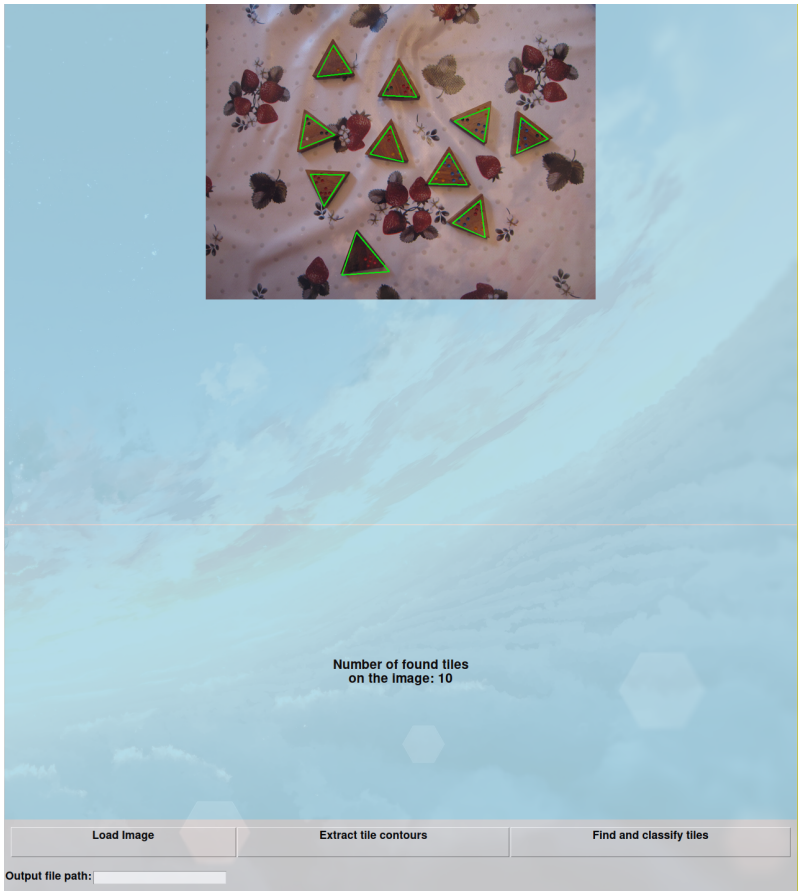

Рис. 10: Интерфейс приложения

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`. 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, так и с целью улучшения качества классификации отдельных фишек. 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. Некоторые из экспериментов имеют хороший потенциал по мнению автора, но не были доработаны в условиях ограниченности времени и ресурсов. Такие эксперименты ниже выделены жирным шрифтом.

- 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.
- **Использование дополнительной информации о цвете маркировки для различных классов**
  Данное исследование зашло в тупик на этапе попытки кластеризации различных оттенков цветов для одного и того же класса маркировки.
  В качестве признаков при кластеризации использовались значения RGB

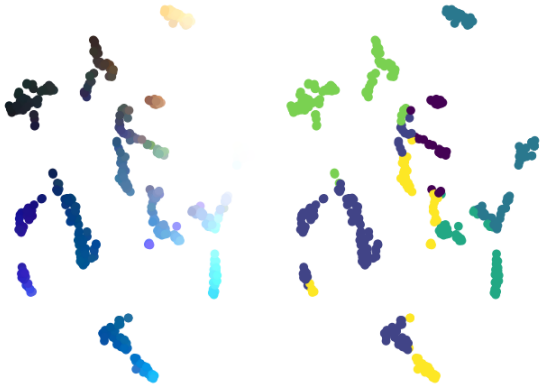

Рис. 11: Кластеризация цветов маркировки

пикселей. Слева изображена визуализация выборки алгоритмом t-SNE, а справа — она же, кластеризованная на 6 классов алгоритмом KMeans. Можно заметить, что как t-SNE, так и KMeans не могут хорошо разделить выборку на 6 классов. Визуально можно убедиться, что это является нетривиальной задачей даже для человека.
Как можно видеть на изображении, из-за большой разницы в освещенности, цвета некоторых фишек из различных классов становятся неотличимы. Таким образом, данный алгоритм мог теоретически являться хорошей

проверкой классификации (для разделения классов с сильно отличающимися цветовыми гаммами). Однако от него пришлось отказаться в силу того, что он может являться только вспомогательным алгоритмом классификации в данной задаче.

- **Использование предположений о количестве точек в маркировке и поиск решения путем проверки гипотез**
  Исследование было продиктовано тем, что мы заранее имеем информацию о количестве точек в каждом из классов. Используя эту информацию, мы можем попытаться подобрать для каждого класса (а, значит, для каждой отдельной гаммы цветов, представляющих этот класс) наиболее эффективные каналы (напомним, что в оригинальном алгоритме используется 12 каналов), а также другие параметры исходя из того, что мы знаем, какого цвета объект, который мы пытаемся выделить. Таким образом, данный эксперимент является идейным развитием предыдущего описанного эксперимента.
  Если подобрать для каждого класса маркировки эффективные параметры, то при обработке каждого изображения достаточно перебрать 6 возможных вариантов параметров и каналов для каждой из гипотез о классе маркировки, после чего проверить, подтвердилась ли гипотеза (т.е. совпало ли количество выделенных объектов с ожидаемым). Если гипотеза подтвердилась, то мы можем с большей уверенностью утверждать, что наш алгоритм нашел верное решение, чем, например, при простой сегментации кругов на изображении.
  Была проведена серия экспериментов по подбору каналов для каждого из классов, однако была выявлена резкая необходимость подбора гиперпараметров для каждого из классов в отдельности. Подбор гиперпараметров и каналов для каждого из классов является довольно трудоемкой задачей, поэтому от этого подхода пришлось отказаться.

- Изучение влияния освещенности на качество сегментации и классификации.
  В результате данного эксперимента появился алгоритм адаптивного осветления изображения вершины фишки на основании средней интенсивности всех ее пикселей, что позволило несколько сгладить разницу в качестве классификации в зависимости от яркости изображения.

Все эксперименты сохранены в файле `Experiments.ipynb`.

6 Выводы

В процессе экспериментов были исследованы различные подходы к сегментации и выделению контуров на изображении. Было проанализировано влияние точечных и пространственных преобразований и их композиций на изображение. Были разработаны и протестированы несколько алгоритмов 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.