

Отчет по лабораторной работе №1

курса «Обработка и распознавание изображений»

Васильев Руслан ВМК МГУ, 317 группа

7 апреля 2021 г.

Содержание

Постановка задачи	2
Описание данных	2
Метод решения	2
Программная реализация	5
Эксперименты	6
Выводы	7

Постановка задачи

Цель лабораторной работы — написать программу для работы с фишками настольной игры «тримино». Этапы работы программы:

1. Считывание входных изображений;
2. Обработка изображений, в которую входят:
 - Сегментация игровых фишек,
 - Подсчет точек, соответствующих вершинам найденных фишек;
3. Вывод в файл координат центров фишек и количества точек в углах;
4. Отображение на экране наглядного изображения, иллюстрирующего сегментацию и подсчет точек.

Кроме того, программа задумывается не только как «черный ящик», но и модуль, из которого можно импортировать функции и настраивать их параметры, улучшая качество сегментации и точность подсчета точек.

Описание данных

Данные, подающиеся на вход, — фотографии фишек, хранящиеся в формате BMP24. Таким образом, мы работаем с трехканальными изображениями, где глубина каждого цвета — 8 бит. Примеры, предложенные в задании, изображены на [рис. 1](#).



(a) Beginner, Pict_1_1



(b) Intermediate, Pict_2_1



(c) Expert, Pict_4_1

Рис. 1: Примеры входных изображений

Три уровня сложности связаны с фоном и освещенностью. В простейшем случае изображение имеет гладкий фон и однородное освещение, в худшем на фоне расположены дополнительные элементы, а фишкы освещены неравномерно. От решения требуется устойчивость к таким особенностям.

Метод решения

Решение разделяется на два ключевых этапа: сегментация фишек и поиск точек внутри найденных объектов. На каждом этапе совершается последовательность

из нескольких преобразований изображения, которую мы продемонстрируем на вырезанном фрагменте из «сложного» уровня (`Pict_3_1.bmp`), [рис. 2a](#). Оставим также координатную сетку для наглядности.

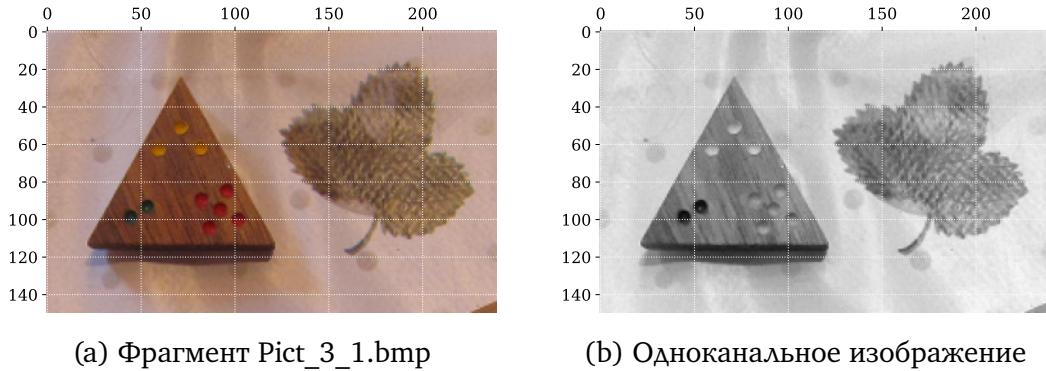


Рис. 2

На первом шаге мы переходим от трехканального изображения к одноканальному. По умолчанию программа оставляет только красный канал ([рис. 2b](#)), поскольку стандартные формулы для перехода в оттенки серого в первую очередь учитывают зеленый, который в нашей задаче не дает хорошую отделимость. Далее в полученном изображении выделяются границы с помощью алгоритма Кэнни ([рис. 3a](#)).

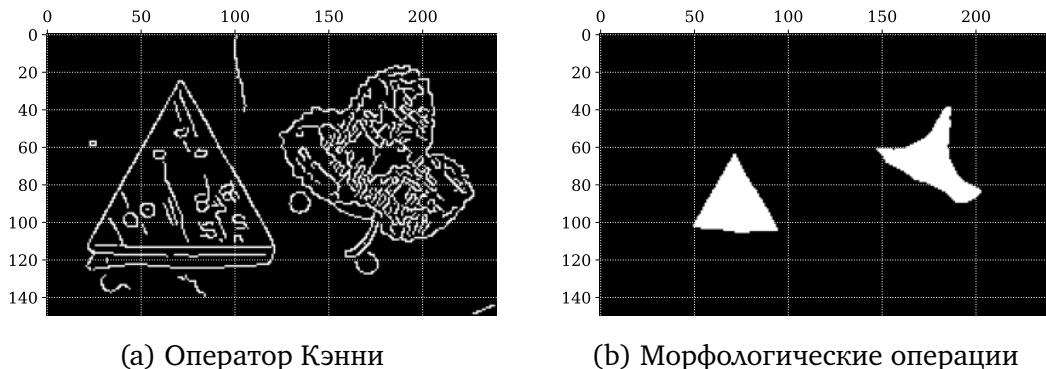
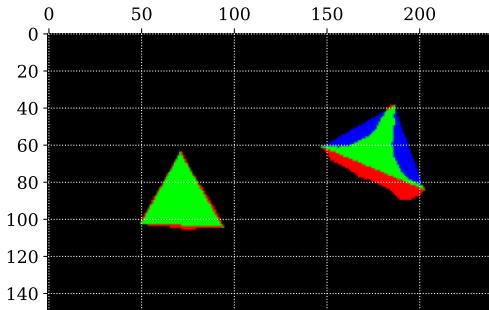


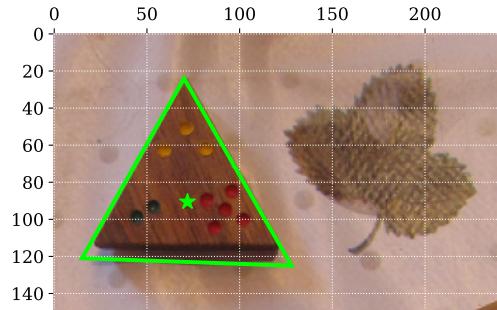
Рис. 3

Обнаружив границы, мы хотим избавиться от шума и всего, кроме треугольников. Применим последовательность морфологических операций: замыкание (чтобы обеспечить полный контур у всех треугольников), заполнение областей и достаточно агрессивная эрозия с кругом в качестве примитива. На выходе ([рис. 3b](#)) мы получим заполненные, но уменьшенные треугольники, а также фоновые объекты, сравнимые по размеру с исходными фишками. Такая последовательность позволяет избавиться от большей части шума и сохранить форму треугольников.

Далее в каждом объекте (связной компоненте) ищутся 3 наиболее удаленных точки, максимизирующих периметр треугольника, который строится на этих вершинах. Но как понять, что мы действительно нашли фишку «тримино»? Для этого используется аналог коэффициента Жаккара для сегментации — IoU (Intersection over Union). В случае двух бинарных изображений эта величина численно равна площади пересечения (того, что мы считаем «положительным» классом, в данном



(a) Иллюстрация IoU

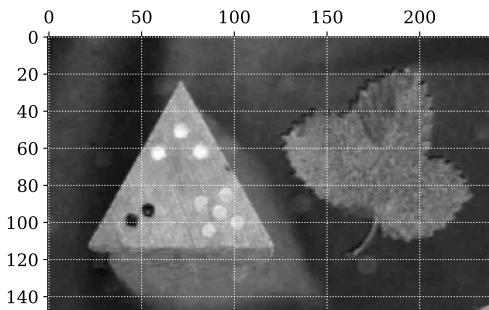


(b) Результат сегментации

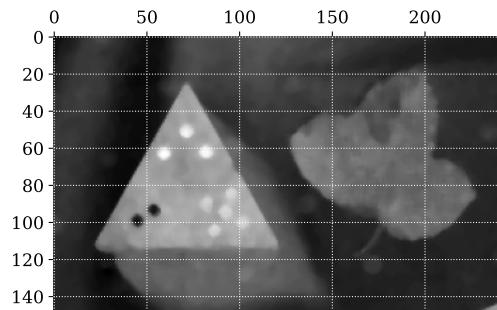
Рис. 4

случае белый цвет), разделенной на площадь объединения. С помощью IoU мы можем оценить, насколько построенный треугольник приближает исходную фигуру. Например, на [рис. 4а](#) для левой компоненты $\text{IoU} \approx 0.88$, для правой — $\text{IoU} \approx 0.43$. Оставляя только объекты с достаточно высоким IoU, мы получаем сегментированные фишками «тримино» с найденными вершинами и центром ([рис. 4б](#)).

Осталось только посчитать число точек в углах найденных треугольников. Основная проблема данного этапа — разные цвета точек. Можно было бы использовать этот факт, например, находя хотя бы одну зеленую точку в окрестности угла, приписывать соответствующей вершине цифру «2». Этот прием неплохо сработает на картинках с хорошим равномерным освещением, но в тривиальном варианте пригоден не для всех предложенных изображений.



(a) Saturation



(b) Медианный фильтр

Рис. 5

Будем бороться с неоднородными цветами другим способом. Для получения одноканального изображения на этом этапе посчитаем насыщенность (saturation) цвета. Сравнивая результат [рис. 5а](#) с [рис. 2б](#), замечаем, что такой подход позволяет хорошо выделять точки, цвет которых по сравнению с древесиной действительно более насыщенный.

Далее применим медианный фильтр ([рис. 5б](#)) и найдем границы с помощью алгоритма Кэнни ([рис. 6а](#)). Причем используя маску сегментированных треугольников, мы можем оставить только то, что находится внутри. Чтобы отличить искомые точки от других элементов (границы фишки, текстуры древесины), мы воспользуемся морфологическими операциями, чтобы заполнить все небольшие полости.

И именно заполненные связные компоненты небольшого размера будут соответствовать нужным точкам. Результат (центр масс каждого найденного объекта) приведен на [рис. 6б](#).

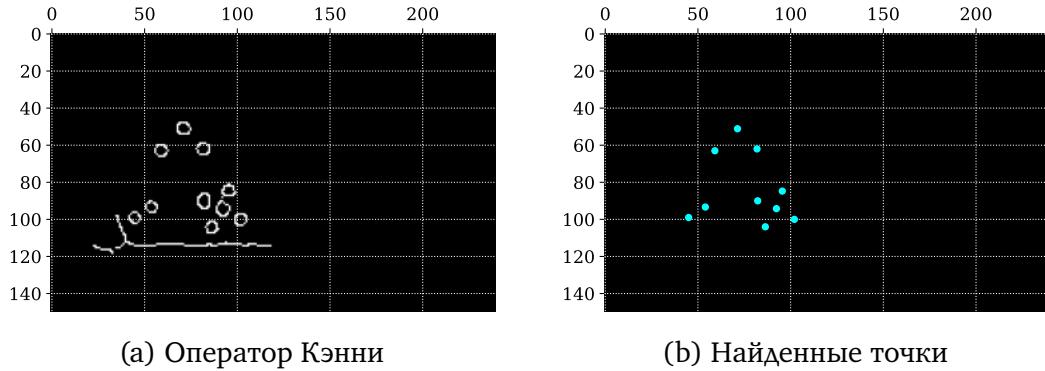


Рис. 6

Остается только отнести точки к нужному углу. Для этого достаточно внутри каждого треугольника для каждой точки найти ближайшую вершину. Таким образом, мы находим все, что требуется в задании. Текстовый выход программы для этого небольшого фрагмента:

```
1
72, 91; 3, 2, 5
```

Таким образом, мы можем эффективно сегментировать фишку «тремино» с помощью последовательности точечных, пространственных, алгебраических и морфологических операций.

Программная реализация

Для простого запуска программы в режиме «черный ящик» достаточно запустить скрипт `trimino.py` с помощью команды

```
python trimino.py input_image
```

где `input_image` — путь к входному изображению. Необходимые библиотеки будут приведены ниже. В процессе работы программа выведет на экран иллюстрацию сегментации и отметит найденные точки. Также иллюстрации будут сохранены в файле `output_figure.png`, а требуемый формат вывода по заданию (число фишек, координаты центров, количество точек по углам) — в файле `output.txt`. В процессе работы программа печатает в терминал текущий этап выполнения. Скриншот с примером работы приведен на [рис. 7](#).

Для более качественной работы возможно импротирание нужных функций из `trimino.py`. Описанные в предыдущем разделе операции настраиваются: например, можно варьировать размеры структурообразующих элементов, параметры алгоритма Кэнни, выбирать метод перехода к одноканальному изображению и т.д.

Как уже ясно из способа запуска, программа написана на языке программирования Python с использованием библиотек:

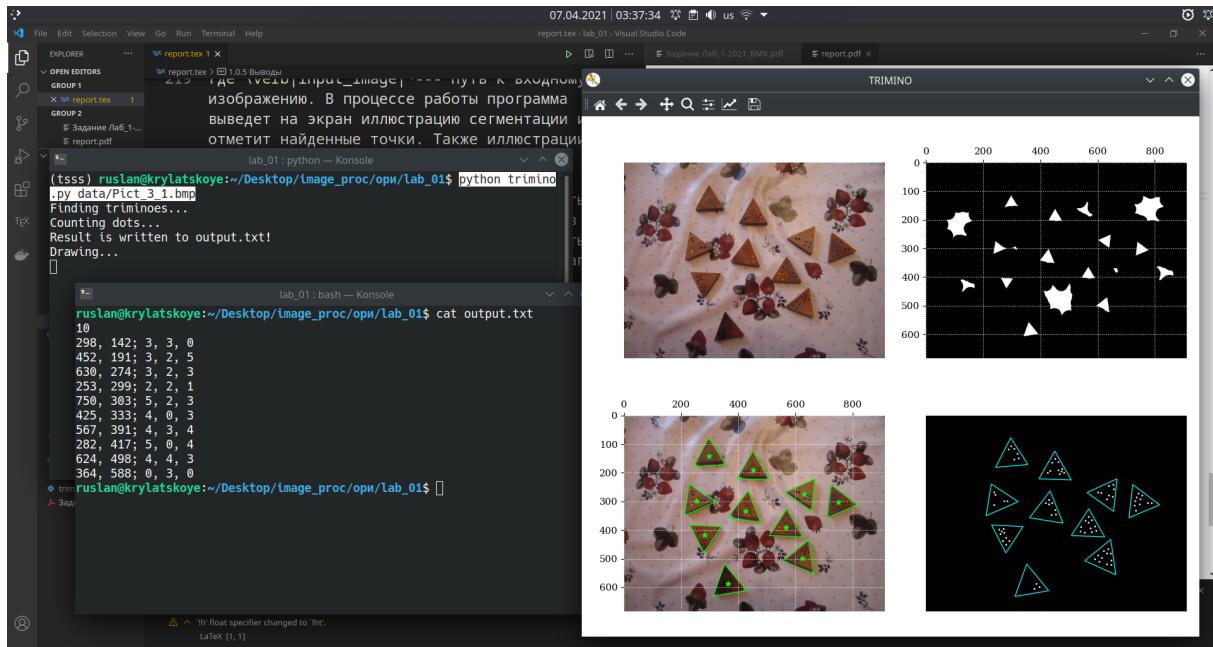


Рис. 7: Скриншот, иллюстрирующий работу программы на изображении Pict_3_1

- `pillow` для ввода-вывода изображений;
- `numpy` для векторных и матричных вычислений;
- `scikit-image` и `scipy.ndimage` для операций с изображениями;
- `matplotlib` для отрисовки графиков.

Эксперименты

Попробуем применить алгоритм к изображениям, приведенным на [рис. 1](#). На первом этапе (сегментация фишек) получим [рис. 8](#).

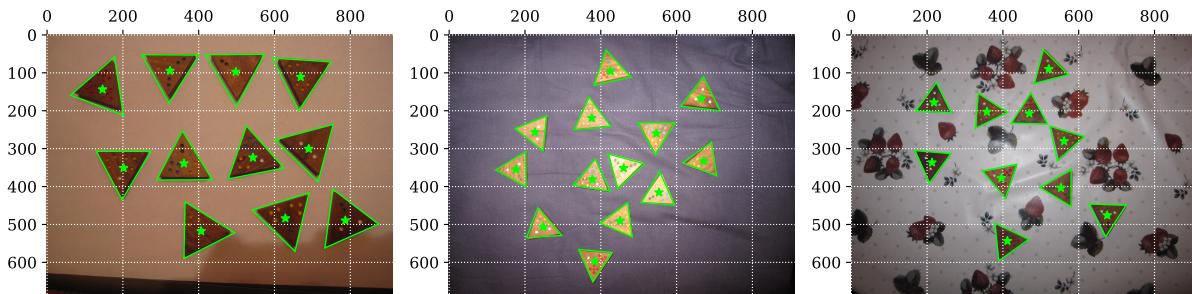


Рис. 8: Сегментация фишек «тримино», три уровня сложности

Алгоритм с высокой точностью сегментирует треугольники. Теперь попробуем посчитать точки в углах фишек.

Выделение точек [рис. 9](#) оказалось более сложной задачей для программы: можно заметить, что точки обнаруживаются не со 100%-й точностью.

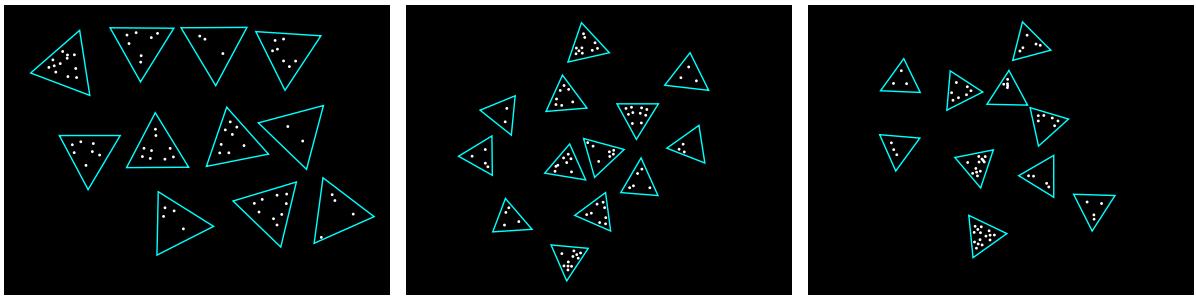


Рис. 9: Подсчет точек на обнаруженных фишках, три уровня сложности

Достаточно сложным элементом является «5», состоящая из красных точек. Из палитры фишек этот цвет наиболее близок к древесине, поэтому он и сливается. Интересно, в задаче обнаружения точек засвеченность фотографий помогает. Благодаря ярким бликам на «Intermediate» и «Expert» пятерки обнаруживаются лучше, чем на тусклом «Beginner».

Выводы

Итак, по итогам данной лабораторной работы получена программа для сегментации фишем «тримино» и подсчета точек на них. Алгоритм, основанный на стандартных операциях с изображениями, можно перенести на другие задачи сегментации примитивных форм. Отметим, что такой подход соответствует обучению без учителя: предоставленные изображения были использованы только для отладки и выбора гиперпараметров по умолчанию. Это определяет недостатки и преимущества метода: для качественной работы на новых данных может потребоваться заново настроить гиперпараметры, но отсутствие процедуры обучения делает использование алгоритма простым и быстрым.