Московский государственный университет имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра математических методов прогнозирования

Отчет по лабораторной работе №1

Илларионова Светлана Владимировна
Группа 317

Москва, 2016

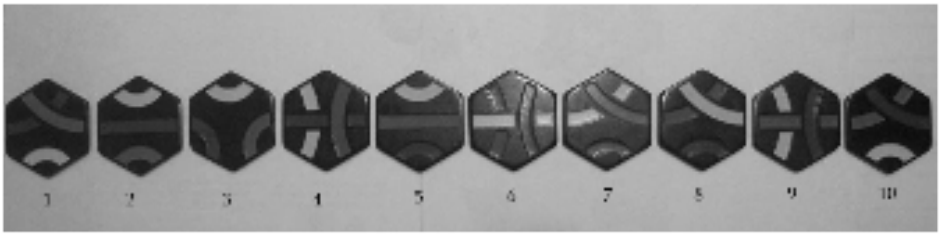
1. Определить количество фишек на изображении

1) **Бинаризация**

Для бинаризации использовалась функция библиотеки OpenCV: cv2.adaptiveThreshold. Адаптивный трешхолдинг позволяет переводит изображение с оттенками серого при неравномерном освещении в бинарное изображение. Исходное цветное изображение было преобразовано в оттенки серого функцией cv2.cvtColor. Параметр $cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C$ задает фильтр Гаусса.

Рис. 1. Исходное изображение

Рис. 2. Изображение после cv2.cvtColor



2) **Выделение контура**

Для выделения контура используется функция: *contours, hierarchy = cv2.findContours(th, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)* Параметр *RETR_EXTERNAL* задает выделение только внешнего контура. *CHAIN_APPROX_NONE* не аппроксимирует точки ребер в отрезки, так как дальше потребуется обход по каждой точке.

3) **Выделение компонент связанности**

Здесь используется функция, которая по полученному контуру закрашивает в заданный цвет (у нас белый) внутреннюю часть фигуры относительно контура.

$$cv2.drawContours(img2, contours, -1, 255, -1)$$

Рис. 3. Бинарное изображение

Рис. 4. Компоненты связности

2. Определить тип и цвет линий на фишке – короткая дуга большой кривизны, длинная дуга малой кривизны, прямолинейный сегмент

1) **Эрозия**

Для определения типа и цвета обрежем края фишек, чтобы при проходе по точкам контура захватывать пиксели, принадлежащие линиям. Воспользуемся cv2.erode(img2, kernel) - эрозия, где первый параметр изображение, второй - ядро, которым будем проходить по контуру. В результате применения данной функции произойдет сужение контура.

2) **Цветовые компоненты связности**

Для каждой фишки выделяем цветовые компоненты связанности, чтобы дальше совершать проход по контуру.

3) **Алгоритм определения типа и цвета линий**

Проходим по контуру, запоминаем встретившийся цвет (проход осуществляется по изображения с выделенными цветовыми компонентами связности). Получаем вектор вида, например: y, y, b, r, b, r, что соответствует короткой желтой дуге и средним синей и красной дугам. Т.е. если дуга короткая, то между двумя ее концами встретятся

Рис. 5. Сужение контура

только пискели, соответствующие черным пикселям фона фишки, если дуга средней длины, то через ребро между двумя ее концами, будет проходить дуга другого цвета. Аналогично для длинной - будут два ребра с дугами другого цвета.

3. Определить номер фишки

Выполнено с использованием результатов предыдущего задания: определяются длины и цвета дуг фишек и сравниваются с исходным изображением, на котором приведена нумерация. Заметим, что фишек с такими одинаковыми параметрами нет, то есть длину и цвет можно считать уникальной характеристикой.

Рис. 6. Red length = 2 Yellow length = 3 Blue length = 2

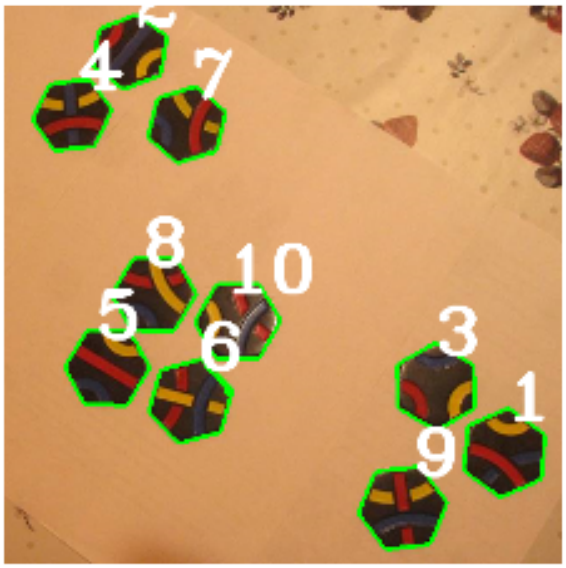
4. Определить расположение и номера всех фишек в кадре

Аналогично предыдущему пункту.

Замечание:

На одном из изображений в кадре находятся предметы, не являющиеся фишками (цветочки на скатерти). При выделение компонент связности, то есть отдельных фишек, проверяется их площадь. Это позволяет отбросить лишние предметы из изображения.

Рис. 7. Много фишек



5. Определить последовательность обхода фишек в мозаике вдоль замкнутого маршрута

Промежутки между фишками убираются применением последовательно эрозии и дилатации. Дальше определяются вершины в бинарном изображении. По ним строятся невыделенные вершины, относящиеся к внутренней части изображения. Это реализуется откладыванием отрезков на заданный угол (60 градусов) с длиной, равной усредненному значению длин отрезков (т.е. ребер) между двумя соседними вершинами.

Рис. 8. Бинарное изображение

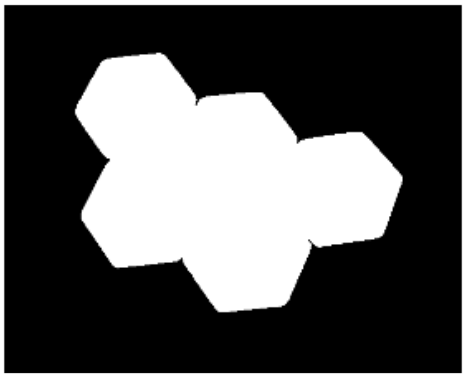
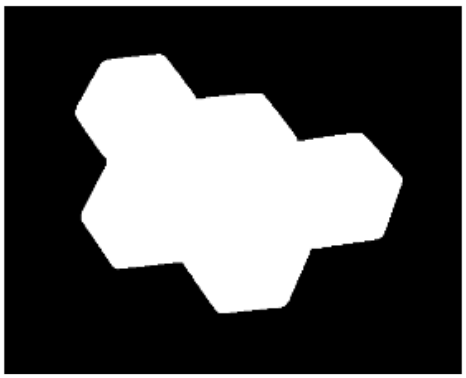


Рис. 9. После применения эрозии и дилатации



Строим граф, вершинами которого являются все вершины фишек, а ребрами — стороны. Путем нахождения в этом графе циклов длины 6, для каждой фишки найдём её вершины в порядке обхода против часовой стрелки. Затем определим номер всех фишек аналогично написанному выше.

Теперь построим граф, вершинами которого являются фишки, и ребро из вершины a и в вершину b существует тогда и только тогда, когда фишка, соответствующая вершине a соприкасается одной из её сторон с фишкой, соответствующей вершине b. Этот граф используется для нахождения искомого пути с помощью перебора всех шести возможных ориентаций для каждой фишки.

Рис. 10. Path: 1 4 9 2 6