Практическое задание
Изучение и освоение методов обработки и сегментации изображений.

Кривонос Анна
317 группа

30 марта 2023 г.

Содержание

1 Постановка задачи

Необходимо разработать программу для работы с изображениями фишек игрового набора Тантрикс. Должен поддерживаться ввод и отображение на экране изображений в формате BMP. Основная задача - классификация фишек в соответствии с заданными номерами, представленными на Рис.1.

Рис. 1:

2 Описание данных

Даннае представляют из себя изображения с фишками формата BMP. Каждая фишка представляет собой правильный шестиугольник черного цвета, на котором изображены сегменты трёх линий синего, красного и жёлтого цветов. Каждая линия имеет определенный тип: короткая дуга малой кривизны, длинная дуга большой кривизны или прямолинейный сектор. Изображения могу содержать либо одну фишку (рис. 2), либо группы из нескольких фишек (рис. 3).

Рис. 2:

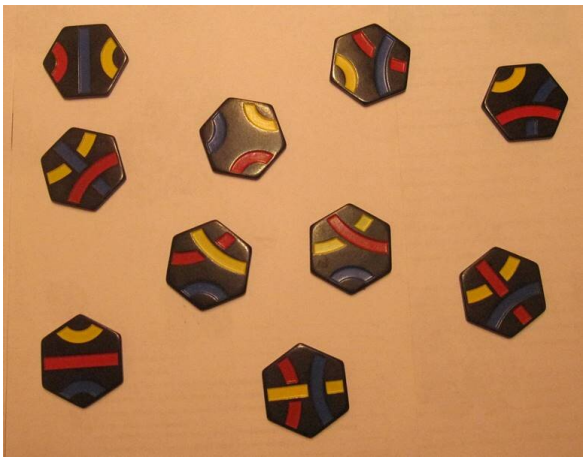

Рис. 3:

3 Метод решения

Для решения задачи применяется следующий алгоритм:

1. Находятся контуры и периметры всех фишек, затем делается апроксимация полученных контуров для вычисления координат 6 углов каждой фишки
2. Если на изображении несколько фишек, то каждая из них вырезается по контурам, полученным в пункте 1. Затем для полученных картинок применяется путкт 1.
3. Находятся центры граней фишки с помощью координат углов, вычисленных выше.

4. Находятся маски каждой дуги по ее цвету, диапазон цветов подбирался вручную.

5. Создаются специальные маски для каждой грани фишки: на черном фоне рисуется белый круг с центром в середине грани и радиусом равным половине длины стороны грани.

6. Премножаются каждая из 6 маскок, полученных на шаге 5(маски с кругом в центре грани) с каждой из 3 масок, полученных на шаге 4(маски цвета) и вычисляется число белых пикселей. В итоге для каждой грани получается количество пикселей каждого цвета в ее окрестности. Идея состоит в том, что вблизи каждой грани больше цвета той дуги, которая к ней примыкает. Таким образом приписываем каждой гране цвет, количество пикселей которого больше. В идеальном варианте должно получиться, что каждому цвету соответствует две грани.

7. На шаге 6 для каждого цвета полученны два номера грани. По эти номерам вычисляется вид дуг:

   - Для каждого цвета находится модуль разности сторон (обозначим r)
   - Если r $>$ 3, то r = 6 - r
   - Затем классифицируется вид дуги по найденному r:
     - r = 1 - короткая дуга малой кривизны
     - r = 2 - длинная дуга большой кривизны
     - r = 3 - прямолинейный сектор

8. Тип и цвет дуг позволяет однозначно определить номер почти всех фишке. Исключение составляют номера 7-8 и 1-10. Для различия этих типов учитывается следующая особеннность: при поиске контуров нумерация граней фишки происходит по часовой стрелке(нумерация с 0). На типе фишек 7-8 синяя дуга имеет малую кривизну, следовательно разность номеров ее граней равна 1, либо грани имеют номера 0 и 5. Найдем максимальный номер i среди номеров синих граней, а если номера 0 и 5, то берем 0. Затем смотрим цвет грани i+1. Если цвет желтый, то тип - 7, иначе 8. Аналогичные действия проводятся для определения типов 1 и 10.

4 Описание программой реализации

Программная реализация находится в файле task_1.ipynb. Для работы с изображениями используется библиотека open_cv.

Для удобства использования программы написанны функции

read_image(path) - чтение изображения с именем path,

show_image(image) - вывод изображения на экран,

draw_contours(image, contours) - возвращает изображение с нарисованными контурами.

Для нахождения контуров на изображении написана функция **find_contours(image)** - возвращает массив контуров на изображении. Она преобразует изображение к черно-белому и находит контуры с помощью функции **cv2.findContours**

Для нахождения углов и периметров фишек реализованна

функция **find_angles_and_per (contours, eps)**, которая аппроксимирует найденные контуры с заданной точностью с помощью функции **cv2.approxPolyDP** и находит периметры контуров. Далее среди всех контуров с помощью периметров выделяются контуры фишек. Алгоритм основан на идее, что периметры всех фишек отличаются не сильно, и других фигур с похожим периметром на изображении нет.

Результат работы функций представлен на картинке:

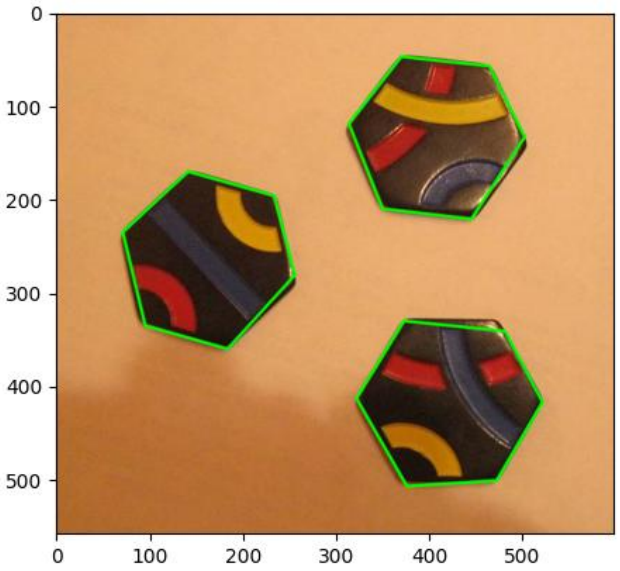

Рис. 4:

С помощью данных функций легко можно вычислить количество фишек на изображении, как количество найденных контуров. Для этого написана дополнительная функция **count_chips(image)**, возвращающая одно число - количество фишек на изображении.

Далее реализованна функция **color_mask(image, h_min, h_max)** которая по заданному изображению строит маску для цветов, попадающий в диапазон [h_min, h_max]. Для этого изображение приводится к типу hsv, где первая компонента h отвечает за цвет. Далее с помощью функции **cv2.inRange** получается нужная маска.

Результат работы функций представлен на картинке:

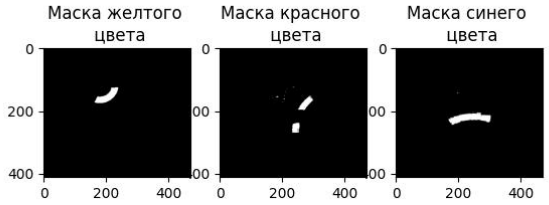

Рис. 5:

Одна из главных функций - **color_and_size(image)** - определяет цвета и типы дуг на изображениях с одной фишкой. С помощью описанных выше функций она находит углы фишки, (если их оказывается больше 6, то находятся ближайшие пары углов и оставляется один из них), вычисляет центры каждой из 6 граней, строит маски цвета. Затем по алгоритму, описанному выше(пункты 6 и 7 алгоритма) вычисляются номера сторон для каждого цвета.

С помощью этой функции написанна функция **print_color_and_size(image)**, которая печатает результаты работы функции **color_and_size(image)** в более удобном для пользавателя виде. Например, для картинки ниже функция выводит ответ : "Желтая дуга малой кривизны, красная дуга большой кривизны, синяя дуга большой кривизны"

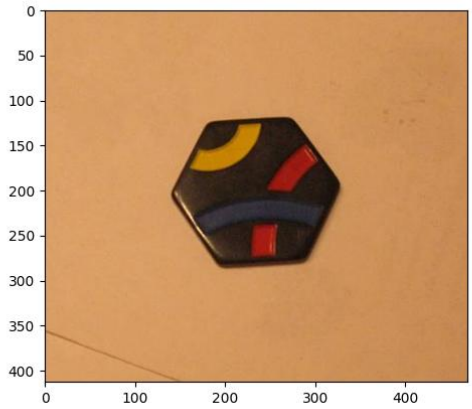

Рис. 6:

Основная функция для распознования типа фишек - **types_chips(image)**. Она находит контуры фишек на изображении, вырезает каждую фишку по контуру и с помощью функции **color_and_size(image)** определяет типы дуг и номера фишек. Наконец, с помощью функции **cv2.putText** пишет номера фишек на изображении и возвращает его.

5 Эксперименты

Протестируем наш алгоритм. Для изображений с одной фишкой полученны следующие результаты:

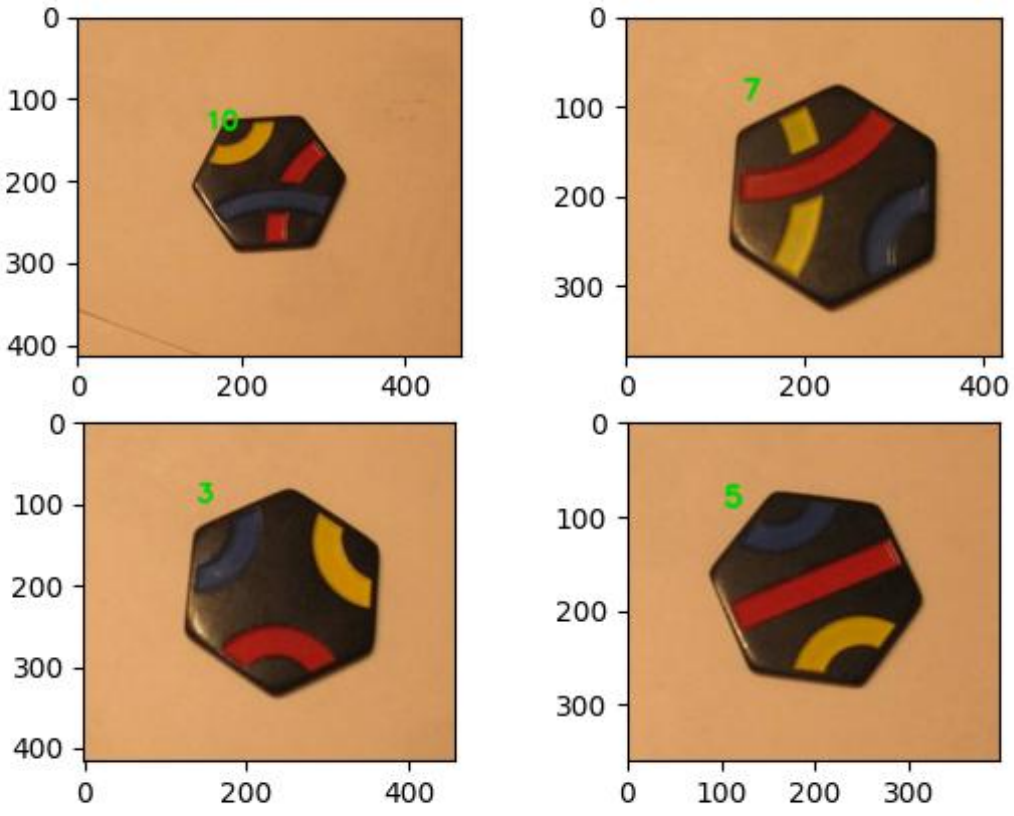

Рис. 7:

Для групп фишек полученны следующие результаты:

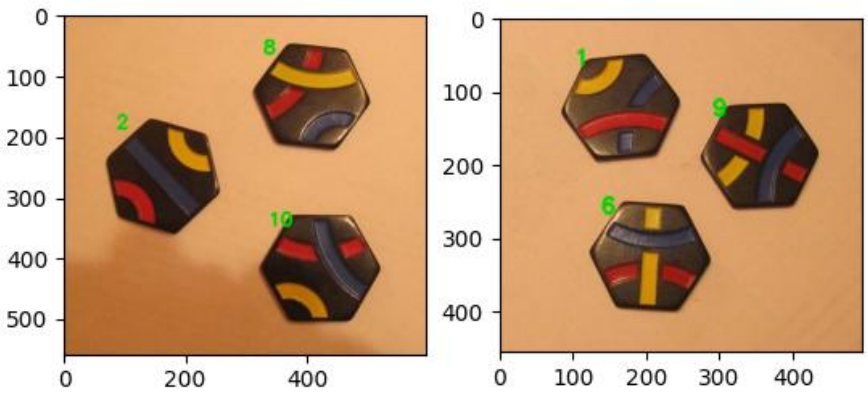

Рис. 8:

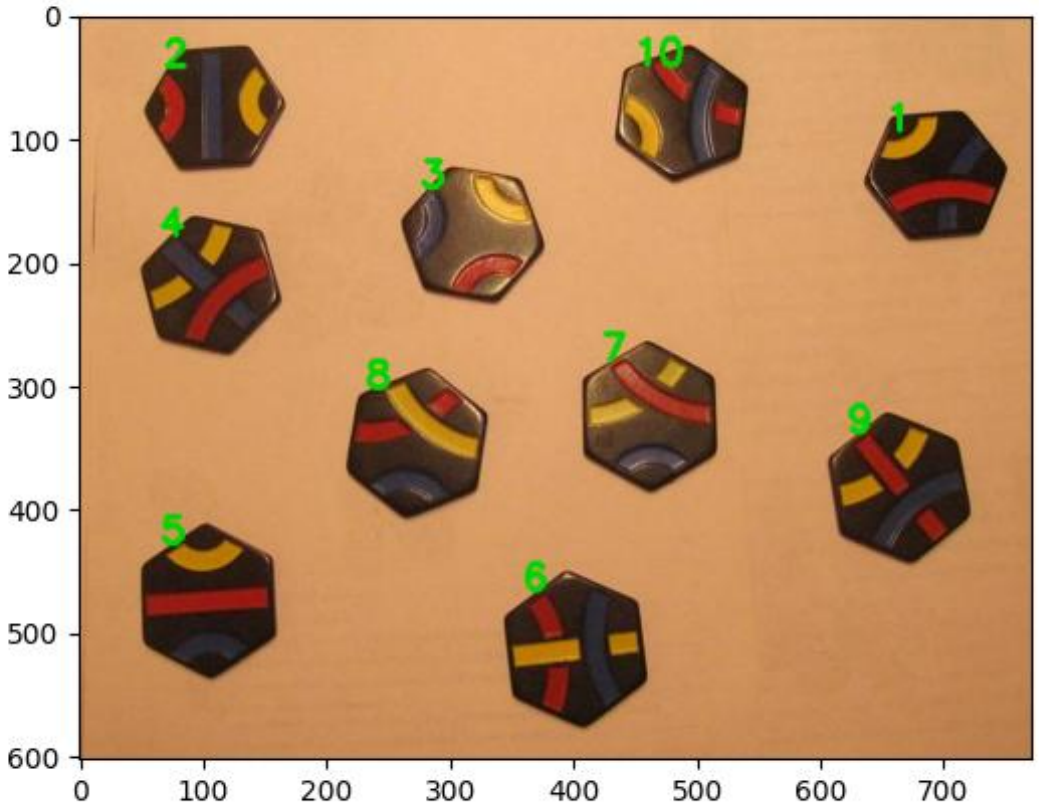

Рис. 9:

Видим, что алгоритм правильно классифицирует фишки для данного набора картинок.

6 Вывод

Была реализована программа, определяющая положение фишек и классифицирующая их в соответствии с заданными номерами. Также, как вспомогательные задачи, были реализованы программы подсчета числа фишек на изображении и описания признаковых характеристик фишек - цвета и типа дуг. Алгоритм показал корректную работу на имеющихся данных.