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

Кафедра ММП, обработка и распознавание изображений

Отчет о выполненном задании «Изучение и освоение методов обработки и сегментации изображений»

Травникова Арина Сергеевна, 317 группа

Введение.

Задание заключается в разработке и реализации алгоритма сегментации изображений и распознавания на них фишек из игрового набора *Puzzle20*. В процессе работы проведены различные эксперименты для сравнения качества алгоритмов с разными параметрами и подбора лучших гиперпараметров. Также выполнено предсказание для изображений из обучающей выборки.

Постановка задачи и описание данных.

В работе описано решение задачи класса Intermediate, которая состоит в сегментации изображения с фишками игрового набора *Puzzle20* на пестром или цветном фоне и определении на нем количества деталей пазла.

Для выполнения задания требуется реализовать алгоритм, на вход которому подается цветное изображение в формате JPG, а выходом является число фишек $k \geq 1$ на нем.

Множество изображений для обучения состоит из фотографий фишек набора *Puzzle20* на однотонном красном или черном фоне, а также пестром в виде тканого полосатого коврика. Все фишки набора имеют свой уникальный рисунок, их контуры состоят из отрезков прямых, дуг окружностей, а также небольших выпуклых и вогнутых частей с округлым краем, которыми детали пазла скрепляются друг с другом.

Описание метода решения.

В решении задачи можно выделить 3 основных шага:

1. Предобработка входного изображения

2. Бинаризация изображения

3. Выделение контуров объектов на изображении и определение, задает ли каждый найденный контур некоторую фишку из набора *Puzzle20*

Далее каждый шаг алгоритма рассматривается более подробно.

1. Предобработка.

Для повышения эффективности работы алгоритма по времени и памяти первоначально выполняется сжатие исходного изображения, поступающего на вход алгоритму.

Так как отделение пестрого фона с изображения представляет основную сложность в задаче, дальнейшие действия по обработке направлены именного на это. Для борьбы с шумом, возникающим от пестрого фона, к цветному изображению применяется размывающий фильтр Гаусса. Далее изображение переводится в полутоновое.

Последующее выполнение операций эрозии и дилатации позволяет удалить с фона большую часть шума и небольшие объекты (мелкие детали самого коврика, узор), а также сгладить границы между деталями пазла и фоном.

Результат работы функций преобразования для изображений с пестрым и цветным фоном представлен на рис. 1.

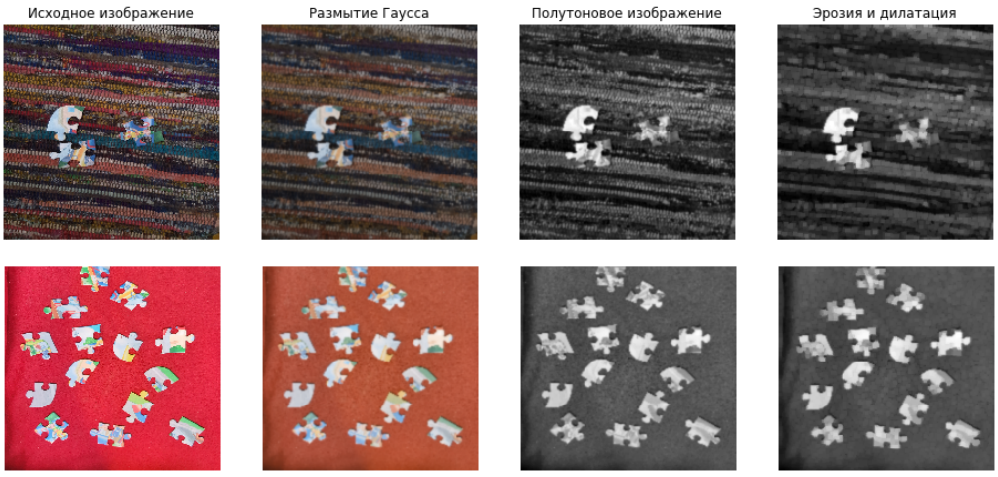


Рис. 1: Результаты последовательного применения описанных выше преобразований к изображениям Motley3.jpg и Red4.jpg из обучающей выборки.

Приведенный пример показывает, что на пестром фоне преобразования действительно хорошо убирают шум, размывают рисунок фона и сглаживают границы и одновременно с этим не портят изображения с цветным фоном, то есть такие преобразования можно использовать для всех фотографий выборки.

2. Бинаризация.

Процесс бинаризации заключается в преобразовании полутонового изображения в двухцветное (бинарное).

Для решения задачи выбран метод глобальной бинаризации с порогом t: все пиксели изображения, яркость которых $i > t$, красятся в белый цвет, остальные – в черный. Наиболее сложным вопросом данного шага является выбор значения параметра t.

Одним из ниболее известных методов подбора порога бинаризации t является подход, основанный на использовании гистограммы яркостей пикселей изображения. На рис. 2 приведены нормализованные гистограммы для изображений с пестрым и цветным фоном из обучающей выборки, а также соответствующие сглаженные гистограммы для них. Красные метки на графиках указывают на подобранный порог бинаризации.

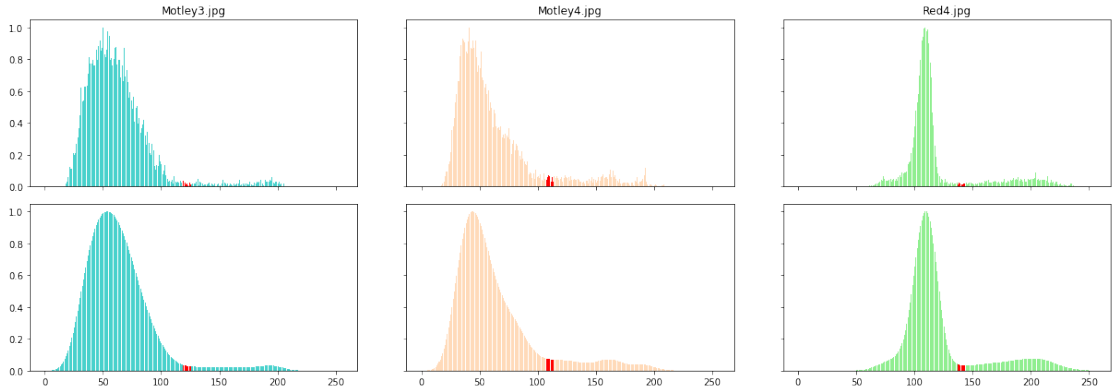


Рис. 2: Нормализованные гистограммы яркости для изображениям Motley3.jpg, Motley4.jpg и Red4.jpg из обучающей выборки и их сглаженные аналоги с указанием порога бинаризации.

При проведении исследования было замечено, что яркость пикселей фона в среднем меньше, чем яркость деталей пазла, и на гистограммах яркости они образуют хорошо выделяющийся горб. Исходя из этого, сделано предположение, что для корректного отделения пикселей фона от фишек $Puzzle20$ достаточно найти по гистограмме правую границу горба, состоящего из элементов бэкграунда, и назначить это число порогом бинаризации.

Для удобства работы с гистограммой выполняется ее ядерное сглаживание. Значение аппроксимирующей гистограммы в точке X определяется формулой:

$$Y(X) = \frac{\sum_{i=0}^{255} K_h(X-i)y(i)}{\sum_{i=0}^{255} K_h(X-i)}, 0 \leq X \leq 255,$$

где X - некоторое значение яркости, $y(x)$ - значение исходной гистограммы в точке x, K - функция ядра с заданной шириной окна h.

В решении используется ядро Гаусса:

$$K_h(x) = \frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}}\big[|x| \le h\big],$$

где величина h является настраеваемым гиперпараметром.

По получившейся сглаженной гистограмме порог определяется как точка, в которой после достижения пика функция перестает убывать или скорость ее убывания становится достаточно мала. Результаты описанной аппроксимации для нормализованной гистограммы яркости и подбора порога бинаризации приведены на рис. 2.

Применяя вычисленный порог для бинаризации изображений, получаем их двухцветный вариант. Примеры на рис. 3.

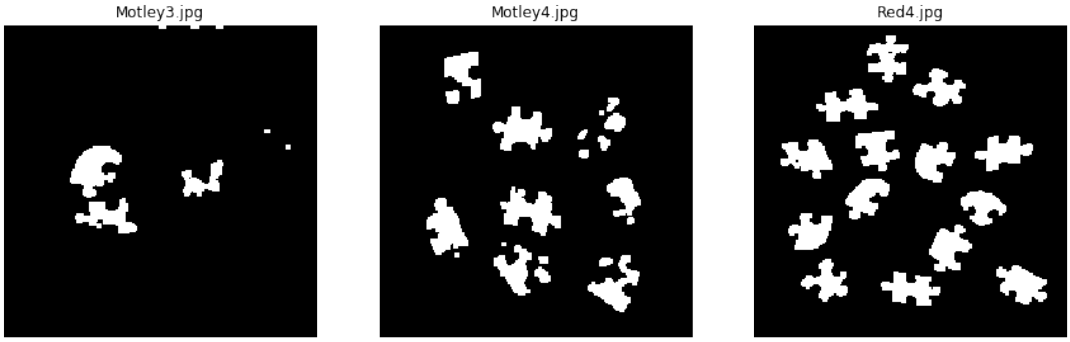


Рис. 3: Бинаризованные изображения Motley3.jpg, Motley4.jpg и Red4.jpg.

3. Выделение контуров.

Следующим шагом алгоритма является распознавание контуров объектов полученного бинарного изображения. При определении, задает ли некоторый найденный контур границу фишки из набра *Puzzle20*, главным признаком является площадь фигуры, которую он ограничивает. После удаления фона с изображения на картинке остаются только элементы пазла большой площади и мелкий шум фона, однако проблема заключается в том, что некоторые фишки оказываются порванными на части.

Для удаления случайных объектов фона сразу удаляются контуры, площадь которых $s < \varepsilon$.

Далее, в предположении, что на изображении всегда есть хотя бы одна фишка *Puzzle20*, будем считать, что фигура максимальной площади всегда соответствует детали пазла. Чтобы правильно вычислить количество фишек на изображении, необходимо учесть, что после бинаризации некоторые элементы пазла

разделены на несколько непересекающихся частей и из них нужно посчитать ровно одну (максимальную по площади), а также удалить оставшийся шум. Будем считать, что текущий контур определяет деталь пазла, если отношение площади фигуры, заключенной внутри него, к найденной максимальной площади фишки больше некоторого заданного числа δ.

На рис. 4 представлены найденные контуры объектов на изображениях обучающей выборки. Белым цветом выделены только те контуры, которые по описанному выше критерию задают фишки пазла.

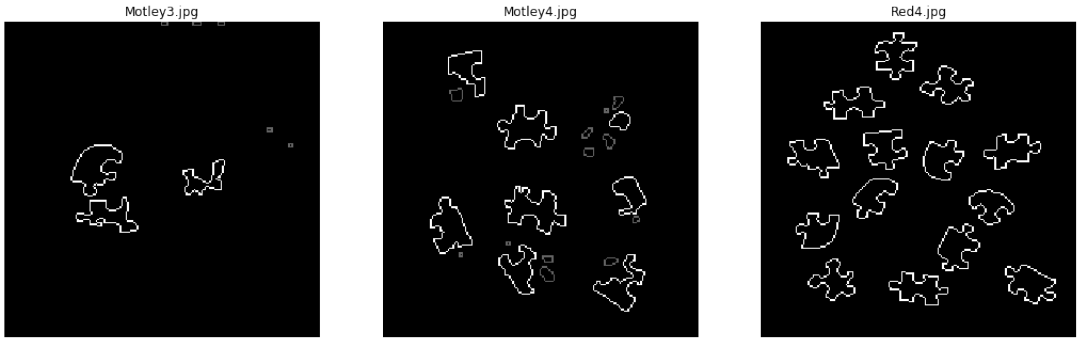


Рис. 4: Контуры объектов изображения Motley3.jpg, Motley4.jpg и Red4.jpg.

Описание программной реализации.

Для программной реализации алгоритма выбран язык Python3 и инструмент Jupyter Notebook для разработки. Работа с изображениями осуществляется посредством библиотек skimage, scipy.ndimge и OpenCV.

Для обработки изображений использованы функции resize, gaussian_filter, morphologyEx (эрозия с последующей дилатацией), cvtColor (перевод в полутоновый формат).

Бинаризация полутоновых изображений осуществялется при помощи функции threshold. Работа с контурами включает в себя 2 основных алгоритма: поиск контуров на бинарном изображении с помощью функции findContours и вычисление площади контура, заданного множеством точек, через contourArea.

Программа, решающая поставленную задачу, состоит из 4 основных функций:

- $transform(image, show = False)$ –
  принимает на вход изображение $image$, считанное в формате BGR,
  возвращает обработанное полутоновое изображение
- $bin\_threshold(image, show = False)$ –
  принимает на вход полутоновое изображение $image$,

возвращает нормализованную гистограмму яркости пикселей , сглаженную гистограмму, и порог бинаризации

- $binarization(image, thresh, show = False)$ –
  принимает на вход полутоновое изображение $image$ и порог бинаризации,
  возвращает бинарное изображение

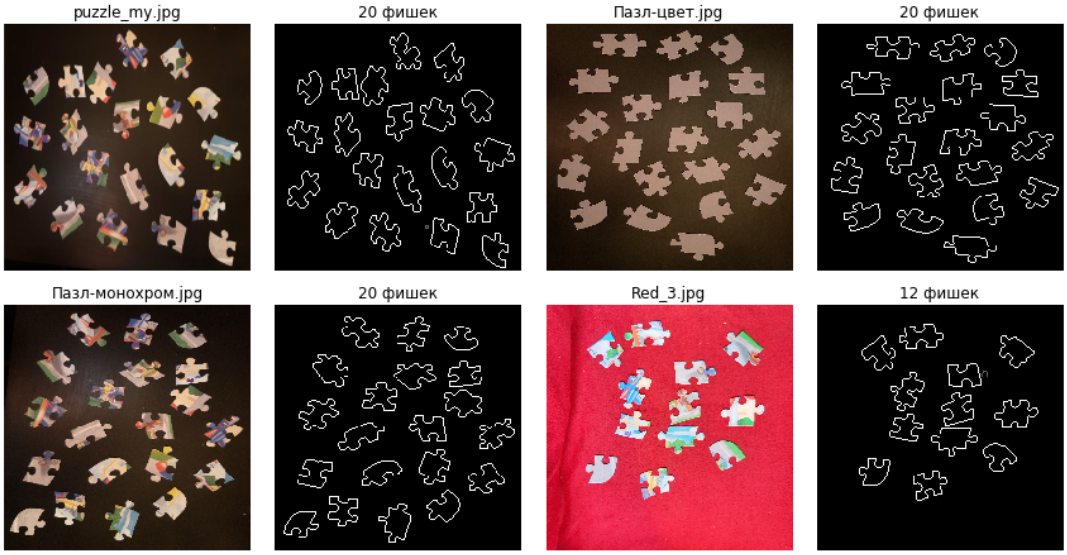
- $num\_puzzles(image, show = False)$ –
  принимает на вход бинарное изображение $image$,
  возвращает изображение контуров фишек и количество деталей *Puzzle20*

Итоговый алгоритм $solve(image\_name, show = False)$ принимает на вход путь к изображению, выполняет его считывание, последовательно запускает описанные выше функции и возвращает результат - число фишек *Puzzle20* на изображении.

Параметр *show* во всех функциях отвечает за то, нужно ли выводить на экран изображение, получающееся в результате работы каждого шага.

Эксперименты.

На рис. 5 приведены результаты работы алгоритма на всех изображениях обучающей выборки. Подписи к исходным фотографиям соответствуют названию изображения, к контурам - распознанному количеству фишек на изображении. Изображение *puzzle_my.jpg* взято дополнительно из файла с заданием.



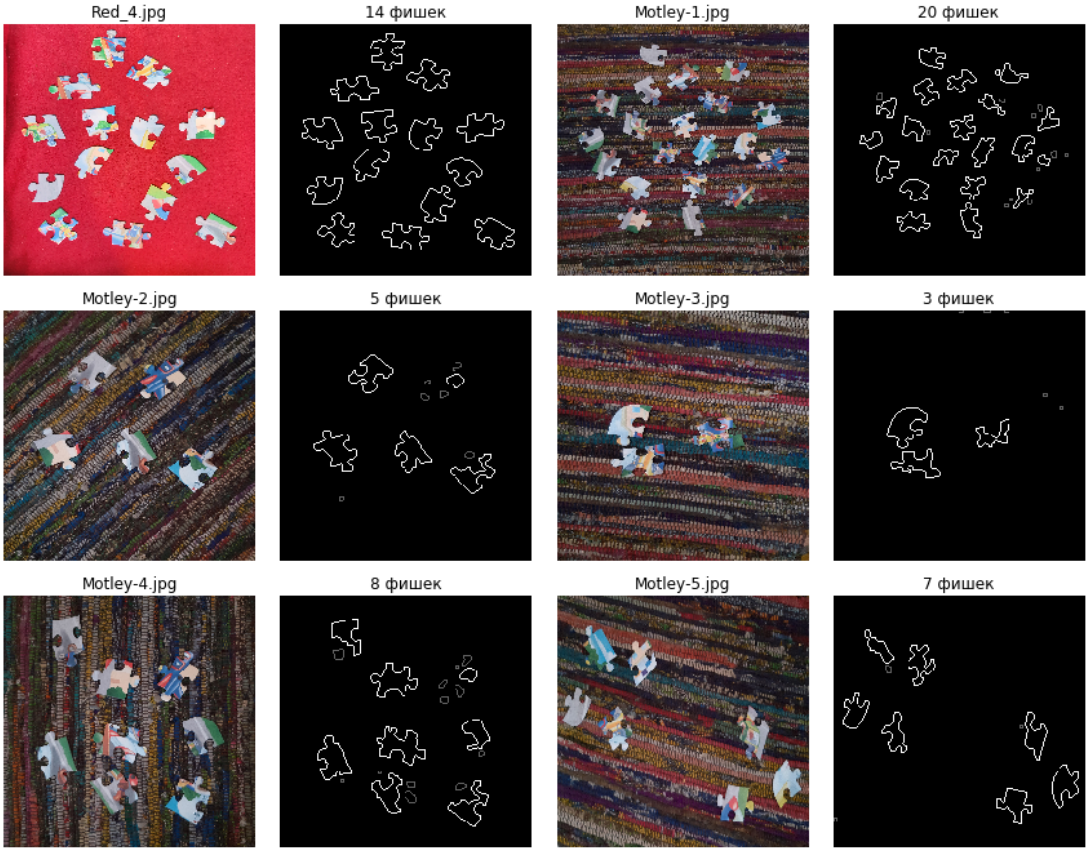


Рис. 5: Результаты работы алгоритма на изображениях из набора для обучения.

Вывод.

Проведенные эксперименты показали, что разработанный алгоритм позволяет правильно оценивать количество фишек пазла на пестром и цветном фоне для всех изображений обучающей выборки.