

Московский государственный университет имени М. В. Ломоносова



Факультет Вычислительной Математики и Кибернетики
Кафедра Математических Методов Прогнозирования

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

«Изучение и освоение методов обработки и сегментации изображений»

Выполнил:
студент 3 курса 317 группы
Суглобов Кирилл Алексеевич

Москва, 2022

Содержание

1 Постановка задачи	1
2 Данные	1
3 Решение задачи	2
3.1 Сегментация карточек	2
3.2 Сегментация фигур на карточках	4
3.3 Распознавание фигур	9
4 Программная реализация	13
5 Эксперименты	16
6 Заключение	16
7 Источники	16

Постановка задачи

Цель работы - разработать и реализовать программу для работы с изображениями карточек игрового набора "Геометрика" которая должна обеспечивать:

- Чтение и отображение входных изображений
- Сегментацию изображений на основе точечных, пространственных и морфологических операций
- Определение числа карточек
- Распознавание фигур на карточках:
 - Выделение фигур
 - Определения типа фигуры: многоугольник или гладкая фигура
 - В случае многоугольника: подсчёт углов и определение его выпуклости
- Вывод маркированного изображения с разметкой PnC или Pn внутри карточки, где P - многоугольник, n - число вершин многоугольника, C - выпуклый многоугольник.

Данные

На вход программе подаётся изображение, фотографии карточек игры "Геометрика" в формате .jpg. Входное изображение имеет 3 канала и 8 бит в каждом из них. Сложность задачи соответствует одному из трёх классов: Beginner, Intermediate, Expert - в зависимости от входного изображения (пёстрый фон, освещённость, чёткость и пр.). На [рис. 1](#) показаны примеры входных данных разной сложности. В изображениях 3 канала, каждый по 8 бит. На карточках изображены геометрические фигуры.



(a) IMG_1.jpg, простой фон, равномерный свет



(b) IMG_7.jpg, простой фон, равномерный свет, но детали размыты



(c) IMG_9.jpg, сложный фон и неравномерный свет (засветы и тени)

Рис. 1: Примеры входных изображений

Решение задачи

Решение делится на 3 основных этапа: сегментация карточек, сегментация фигур на карточках и распознавание фигур (выделение многоугольников, их число вершин и выпуклость). Рассмотрим последовательно все этапы решения с примерами разной сложности.

Сегментация карточек

На простом фоне даже при неравномерной освещённости карточки хорошо отделимы на канале насыщенности - saturation. После применения оператора Кэнни для обнаружения границ $\sigma = 0.5$ выделяются границы карточек и внутренняя текстурированная площадь карточек.

Далее, применяя морфологические операции, могут быть получены маски карточек. Это алгоритм сегментации карточек по границам, шаги которого рассмотрены на [рис. 2](#) на примере одного входного изображения с простым фоном.

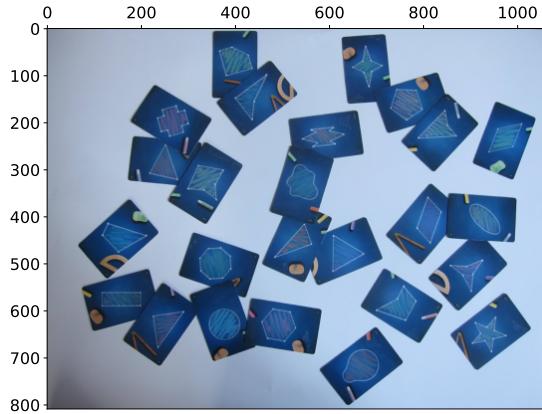
Однако на изображениях с пёстрым фоном алгоритм сегментации по границам ожидаемо показывает плохой результат: алгоритм Кэнни выделяет и объекты на фоне. И в результате вышеупомянутых морфологических операций, как видно на [рис. 3](#), маска карточек получается сильно избыточной: покрывает больше, чем нужно.

Важно, что сегментация по границам в результате даёт покрытие площади всех карточек: необходимое, либо избыточное, но **не меньшее**. Поэтому для сегментации карточек на любых изображениях можно использовать сначала сегментацию по границам: хуже от этого не будет, соблюден принцип «не навреди» из медицинской этики.

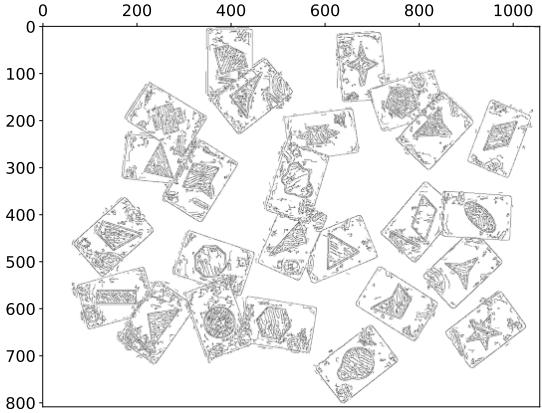
Будем строить сегментацию карточек шагами, на которых мы не теряем (не обрезаем карточки), а либо приобретаем (уточняем маску), либо оставляем маску карточек неизменной, разумно воспользоваться ещё одним принципом - максимой римского сената «разделяй и властвуй». А именно: получим итоговую маску карточек как бинарное произведение масок, независимо полученных разными методами «без потерь». В результате после произведения останутся только общие площади, то есть эти независимые методы должны перекрывать недостатки (избытки площади на результатах) друг друга.

Заметим, что на входных изображениях всегда достаточно синего цвета - цвета внутренней площади карточек. На [рис. 4](#) приведены каналы hue рассматриваемых изображений: с простым и с пёстрым фоном - и гистограммы для них. Именно в канале hue, в отличие от всех других каналов в схемах RGB и HSV, выраженная отделенность пика, отвечающего за синий цвет карточек. Так же видно, что в канале hue выделяются посторонние элементы на самих карточках. Сегментируем карточки (без посторонних элементов на них) с помощью цвета.

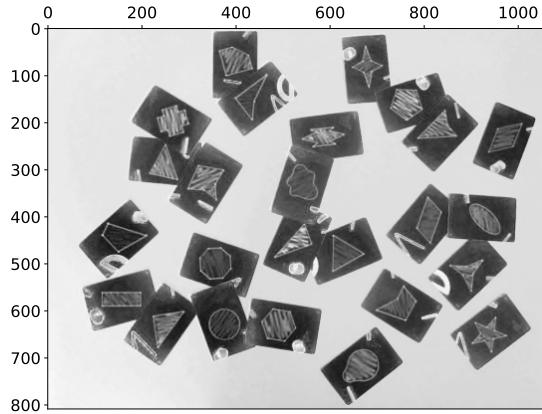
Обрежем лишние цвета на гистограмме. По собственному эвристическому алгоритму



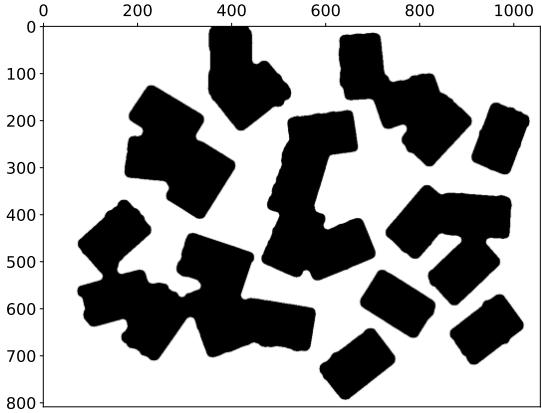
(a) IMG_4.jpg, оригинальное изображение
(простой фон, неравномерный свет)



(c) IMG_4.jpg, бинарное изображение: оператор
Кэнни



(b) IMG_4.jpg, 1-канальное изображение:
насыщенность (saturation)



(d) IMG_4.jpg, бинарное изображение:
морфологические операции

Рис. 2: Сегментация карточек по границам, простой фон

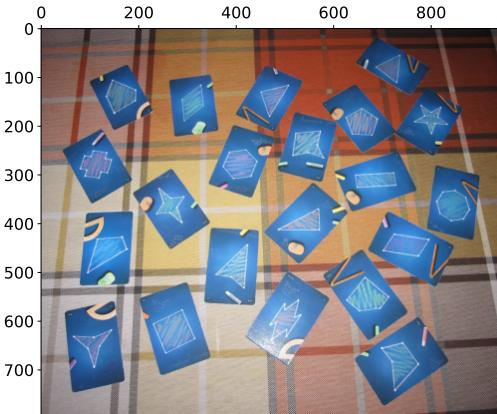
клиппинга гистограммы: ищется максимальный пик и по обе стороны от него (в меньшую и в большую) ведётся поиск точек, значения в которых равны некоторому порогу (threshold), по умолчанию задаваемому как

$$thr = median + factor * |mean - median|, \text{ где по умолчанию: } factor = 0.4$$

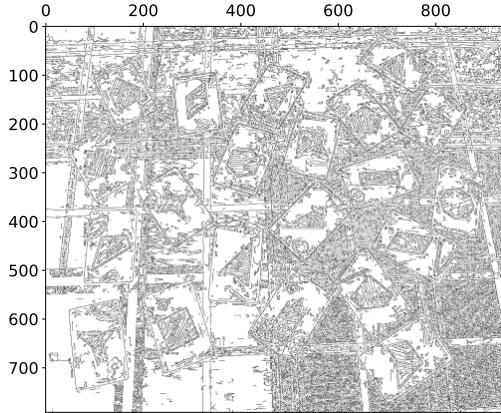
То есть по умолчанию это значение между медианой и средним арифметическим распределения яркости на изображении. Медианные значения явно отделяют пики друг от друга, и если взять чуть больше медианы, получим неплохой порог для их разделения. На гистограммах [рис. 4](#) отмечены соответствующие точки и значения. Рассмотренный порог для плотности яркостей на границах пика даёт два других порога - для бинаризации изображения: выделяются все точки максимального пика. Результаты можно видеть на [рис. 5](#).

Исходя из бинарных масок максимальных пиков, снова получен алгоритм «без потерь», который уточняет маску, но не обрезает карточки, - сегментация по цвету. На пёстром фоне алгоритм показал хороший результат: границы выделены достаточно хорошо и в качестве бонуса получаем отсутствие !посторонних элементов! на самих карточках: их цвет не входил в пик, поэтому помехи были удалены. А на изображении с простым, синим фоном, результат плохой - обрезалось только затемнение в левом нижнем углу, потому что остальная часть изображения почти полностью лежит в максимальном пике, как видно по [рис. 5](#).

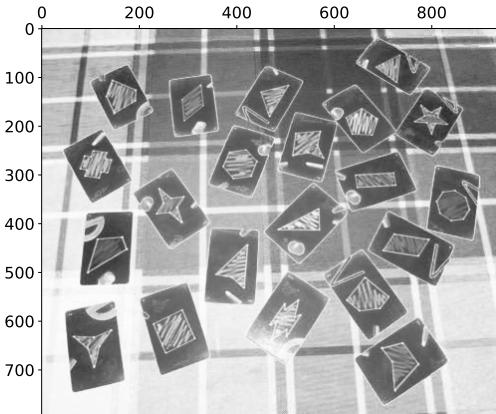
Но, главное, что в обоих случаях не обрезалось ничего лишнего (необходимые площади карточек без посторонних элементов на них, покрыты).



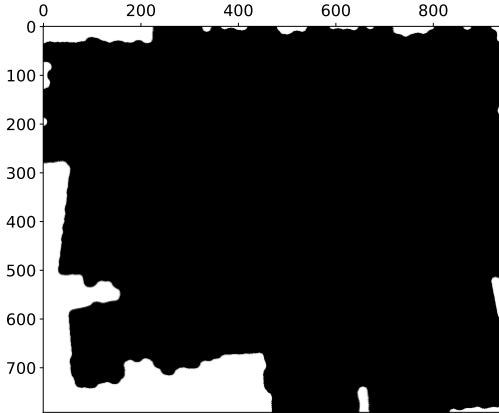
(a) IMG_9.jpg, оригинальное изображение
(простой фон, неравномерный свет)



(c) IMG_9.jpg, бинарное изображение: оператор
Кэнни



(b) IMG_9.jpg, 1-канальное изображение:
насыщенность (saturation)



(d) IMG_9.jpg, бинарное изображение:
морфологические операции

Рис. 3: Сегментация карточек по границам, пёстрый фон

Тогда перемножим маски, полученные при сегментации по границам и при сегментации по цвету.

На [рис. 6](#) видно, что маски карточек достаточно точные, но в случае простого фона не убраны помехи на карточках. Так как почти всё изображение находится в одном пике, то, как видно на [рис. 4](#), медиана и среднее арифметическое, а, следовательно, и порог *threshold* - низкие. Поэтому плотности яркостей помех включаются в максимальный пик. В этом случае необходимо: обрезать изображение в канале *hue* полученной маской, на гистограмме поро- странство вне маски (чёрный цвет), поднять *threshold* (по умолчанию он поднимается до 0.5) и снова бинаризовать максимальный пик.

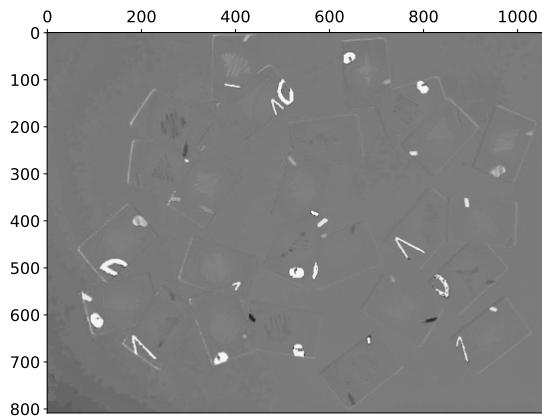
Повторная сегментация по цвету изображения, обрезанного маской, показана на [рис. 7](#).

В результате повторной сегментации цветом с карточек изображения с простым фоном удаляются помехи, а маска карточек изображения с пёстрым фоном почти не меняется.

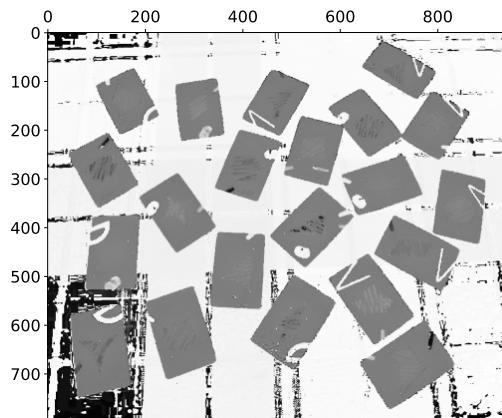
Итого, сегментация карточек без помех на них завершена, маска получена.

Сегментация фигур на карточках

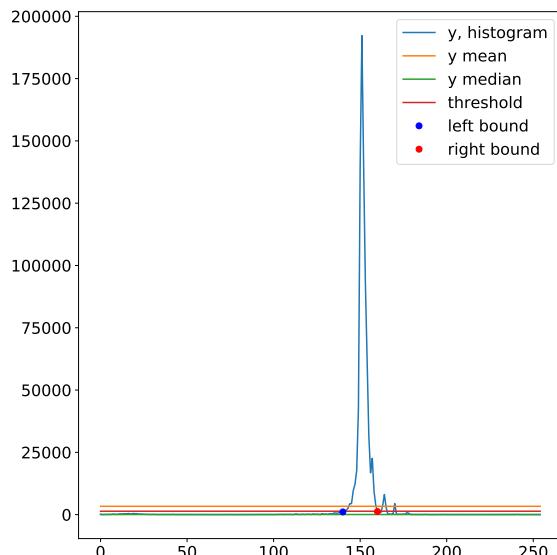
Посмотрим на вырезанные карточки (почти) без посторонних элементов в полутонахом канале *gray* на [рис. 8](#). Видно, что маски получились хорошие. Теперь нужно сегментировать сами фигуры. Везде границы примерно различимы, только на втором изображении с пёстрым фоном и неравномерным освещением на одной из карточек пятно на части фигуры. Это



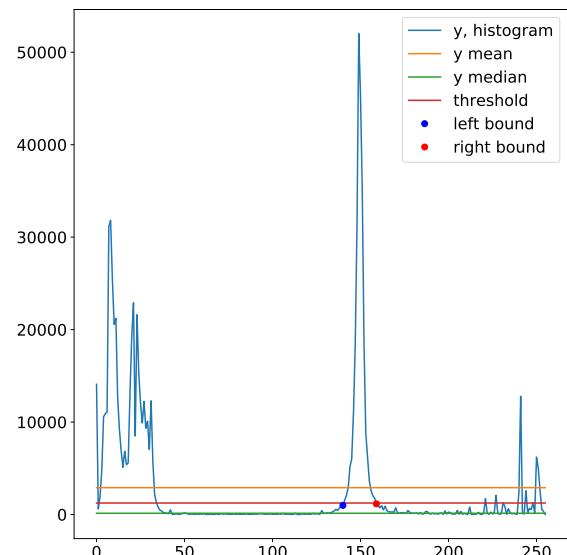
(a) IMG_4.jpg, 1-канальное изображение:
оттенок (hue)



(b) IMG_9.jpg, 1-канальное изображение:
оттенок (hue)

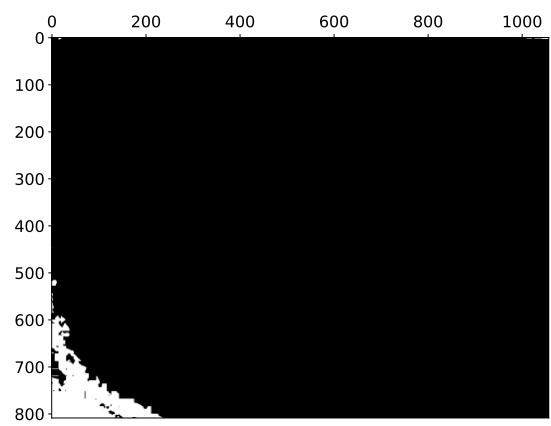


(c) IMG_4.jpg, гистограмма канала hue

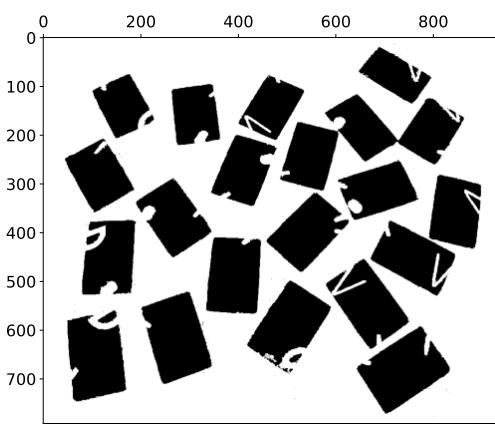


(d) IMG_9.jpg, гистограмма канала hue

Рис. 4: Сегментация карточек по цвету, гистограммы для простого и для пёстрого фона

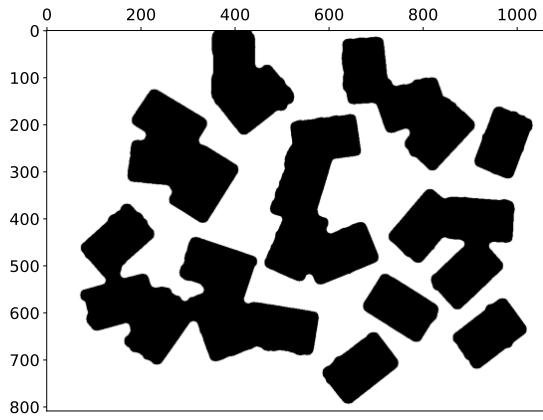


(a) IMG_4.jpg, бинарное изображение: пик
яркости на канале hue

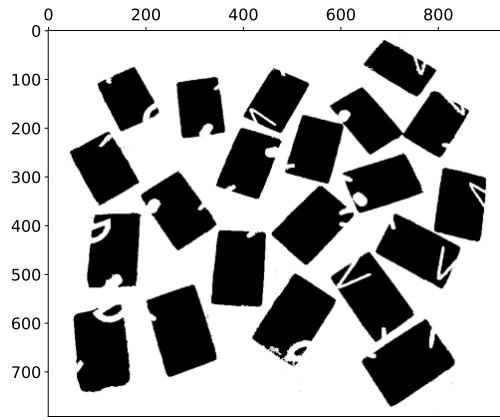


(b) IMG_9.jpg, бинарное изображение: пик
яркости на канале hue

Рис. 5: Сегментация карточек по цвету для простого и для пёстрого фона



(a) IMG_4.jpg, бинарное изображение: маска карточек



(b) IMG_9.jpg, бинарное изображение: маска карточек

Рис. 6: Сегментация карточек по границам и по цвету для простого и для пёстрого фона

тёмное пятно на полутоновом изображении - засвет на оригинале, который из всех каналов яснее всего проявляется в канале saturation (на других каналах граница в этом месте хуже).

То есть сам засвет "находится" в канале saturation, а так же в этом канале самые выраженные границы у фигур, поэтому будем работать с каналом saturation. Избавимся от неравномерного освещения путём деления изображения в канале saturation на его же размытую копию $gaussian(\sigma = 1.75)$, избавляясь таким образом от низкочастотной компоненты. Частное от деления будет низкоконтрастным - эквализуем его. Результат показан на [рис. 9](#).

В результате этих преобразований удалось устранить некоторые дефекты освещения. В месте, где был засвет, есть немного шума, тем не менее граница различима. И в целом результат получился достаточно шумным, а границы в некоторых местах имеют чуть более тёмные пиксели, что будет давать разрывы при бинаризации. Нужно сделать границы ещё более чёткими и яркими, отличными от окружающего пространства.

Для решения этой проблемы применялся ряд последовательных операций над эквализированным изображением: размытие $gaussian(\sigma = 1.0)$, нерезкое маскирование с $radius = 25$, $amount = 3$ и снова размытие $gaussian(\sigma = 1.0)$. В результате границы стали толще и ярче, чётче, а множественные шумы размылись и не так выделяются. Нерезкое маскирование проводилось по формуле:

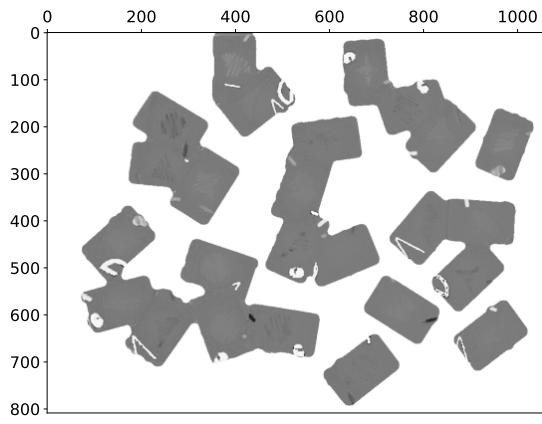
$$sharpened = original + (original - blurred(radius)) \times amount$$

Теперь, если бинаризовать полученное изображение методом Оцу, получится изображение с качественными границами у фигур, помехи от которых находятся на расстоянии за счёт «чистого» фона карточек. Самое время взять от полученного бинарного изображения только внутренности карточек, без границ - перемножим полученный результат с подготовленной (операция закрытия для устранения небольших возможных дефектов и операция эрозии для уменьшения, то есть для исключения границ) маской карточек.

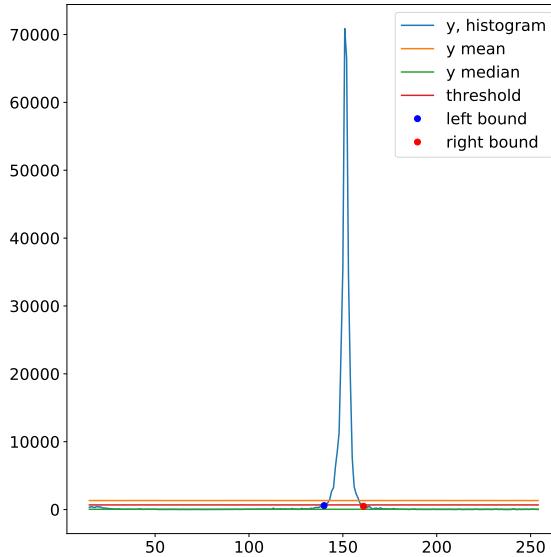
Как видно из [рис. 10](#), в результате - чёткие контуры и внутренняя штриховка фигур, но вокруг шумы.

Для борьбы с такими близкими и сложными шумами применялась сложная последовательная ступенчатая цепочка морфологических операций:

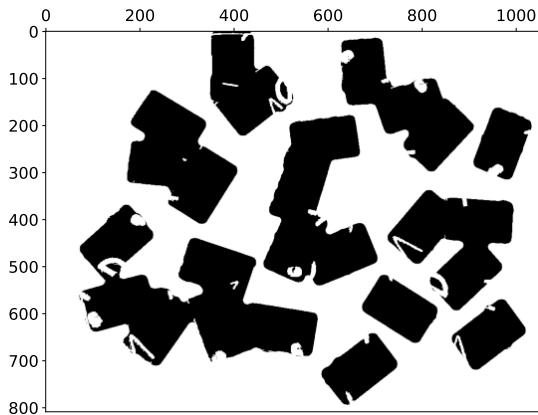
1. Закрытие, заполнение контуров, удаление небольших объектов
2. Закрытие, заполнение контуров, открытие, удаление небольших объектов
3. Закрытие, заполнение контуров



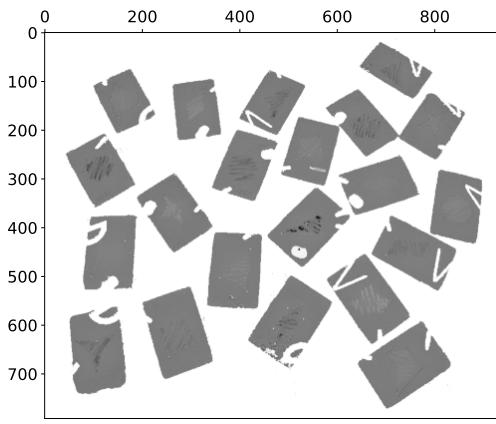
(a) IMG_4.jpg, 1-канальное изображение:
карточки в канале hue



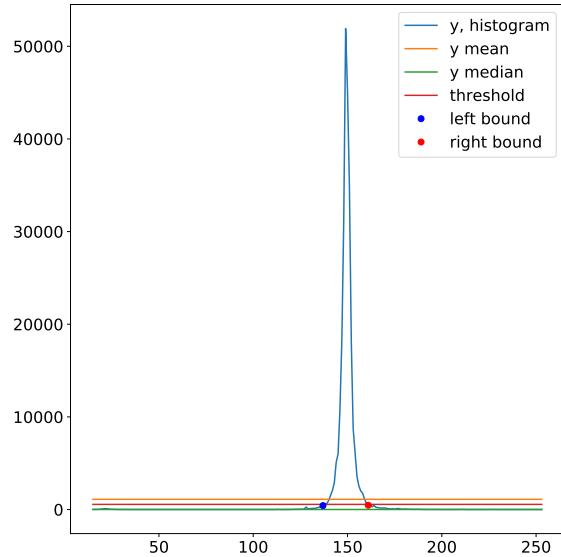
(c) IMG_4.jpg, обрезанная от яркости 15
гистограмма канала карточек в канале hue



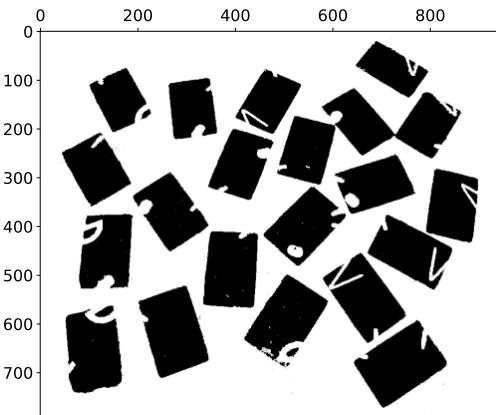
(e) IMG_4.jpg, бинарное изображение: итоговая
маска карточек без посторонних элементов



(b) IMG_9.jpg, 1-канальное изображение:
карточки в канале hue



(d) IMG_9.jpg, обрезанная от яркости 15
гистограмма канала карточек в канале hue

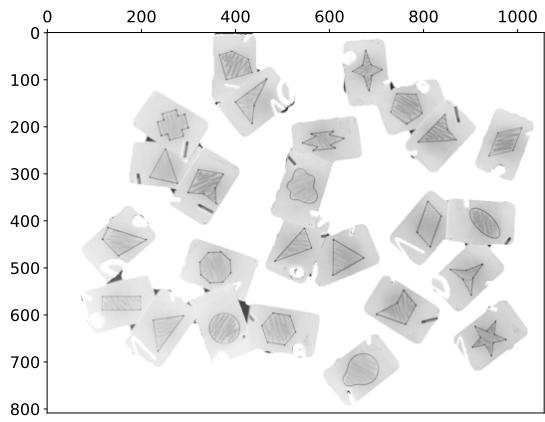


(f) IMG_9.jpg, бинарное изображение: итоговая
маска карточек без посторонних элементов

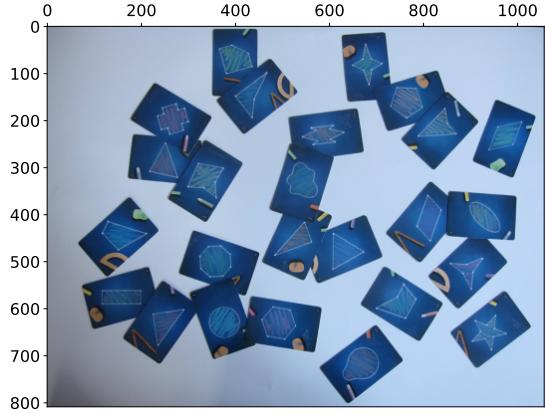
Рис. 7: Сегментация карточек по границам и по цвету для простого и для пёстрого фона

Шаги продемонстрированы на [рис. 11](#).

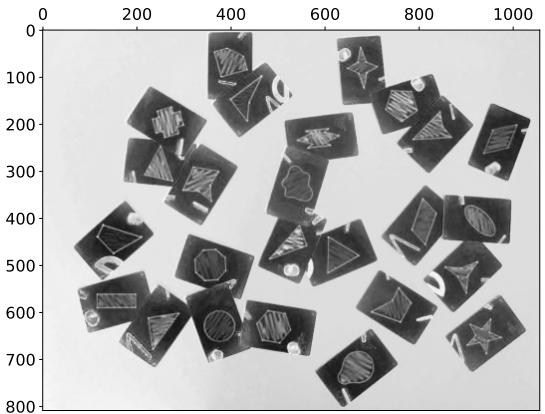
Зыкрытие и заполнение контуров делалось для формирования заполненных фигур с закрытыми контурами. Только после этого удалялись небольшие объекты - шумы. Ведь, если



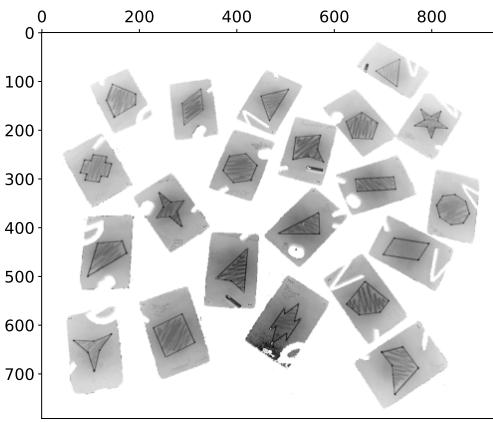
(a) IMG_4.jpg, 1-канальное изображение:
вырезанные карточки, канал gray



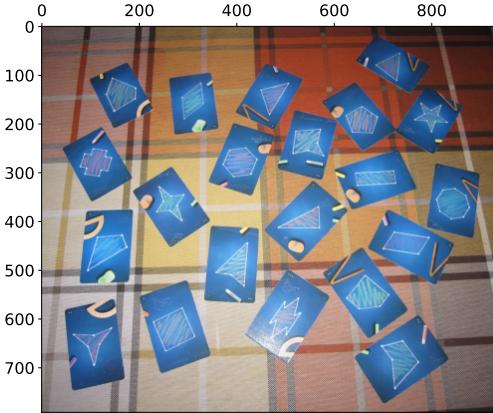
(c) IMG_4.jpg, оригинал



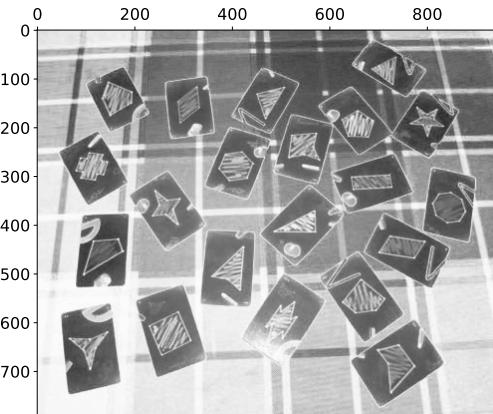
(e) IMG_4.jpg, 1-канальное изображение:
насыщенность (saturation)



(b) IMG_9.jpg, 1-канальное изображение:
вырезанные карточки, канал gray



(d) IMG_9.jpg, оригинал

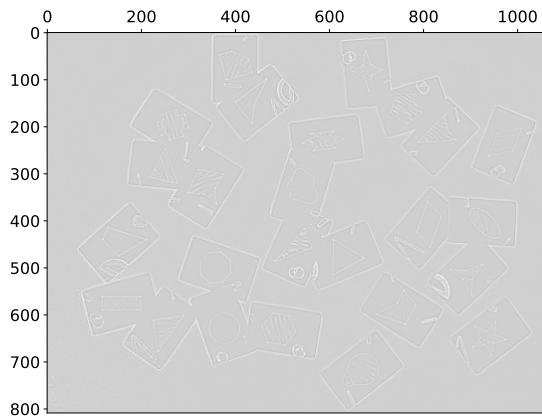


(f) IMG_9.jpg, 1-канальное изображение:
насыщенность (saturation)

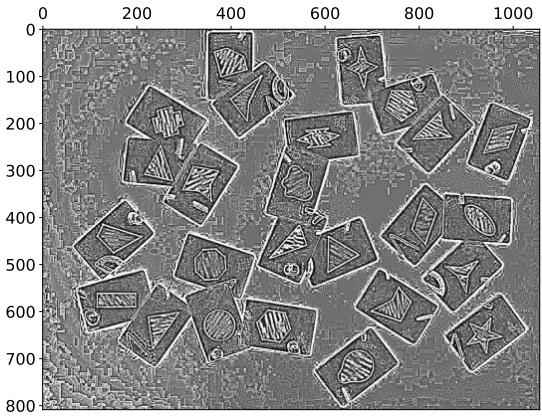
Рис. 8: Неравномерное освещение на карточках

не сделать предварительные закрытие и заполнение, то удалились бы и части фигур. Причём, масштабы закрытия и заполнения возрастают, потому как, если сразу сделать сильное закрытие, фигуры будут объединяться вместе с ещё не убранными шумами, создавая объект, который более не разделить. Открытие уменьшает шумы и фигуры, но в первую очередь - шумы, поэтому удаление шумов постепенно уменьшает свой масштаб. Назовём полученную маску **предфинальной**.

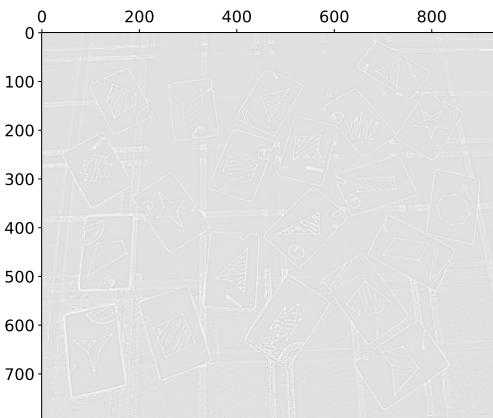
После цепочки этих морфологических операций, создаётся **дополнительная маска** - копия



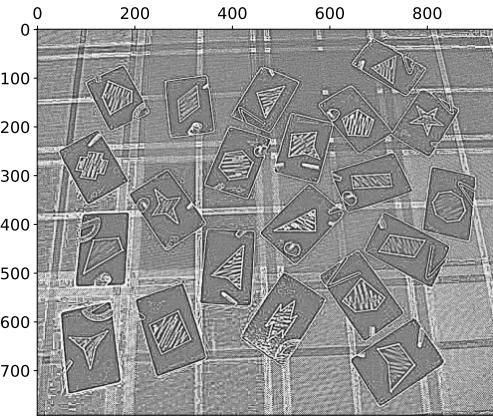
(a) IMG_4.jpg, saturation, частное от деления



(c) IMG_4.jpg, эквализованное частное от деления



(b) IMG_9.jpg, saturation, частное от деления



(d) IMG_9.jpg, эквализованное частное от деления

Рис. 9: Неравномерное освещение на карточках

предфинальной - к которой применяется сильная операция открытия и удаления небольших объектов. После чего дополнительная маска увеличивается дилатацией до минимальных размеров, чтобы вместить в себя предфинальную, и они перемножаются. Сделано это для того, чтобы возможные незначительные шумы на предфинальной маске, которые не удалось убрать цепочкой морфологических операций, отделить операцией открытия и удалить или просто уменьшить. А после перемножения масок получится более ограниченное множество, которое уже не будет содержать этот шум. Это показано на [рис. 12](#)

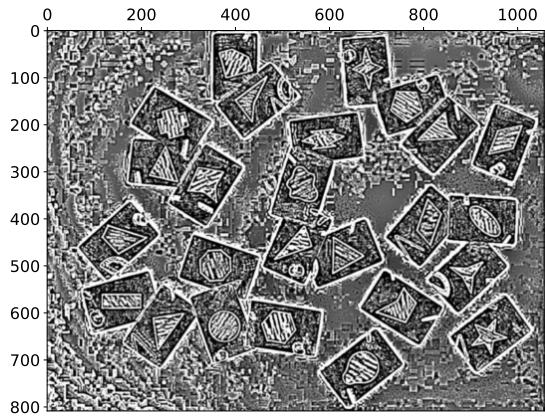
В результате получаем **финальную маску** фигур.

Заметим, что все описанные преобразования универсально применяются для изображений как с равномерной, так и с неравномерной освещённостью и в итоге дают качественную маску фигур.

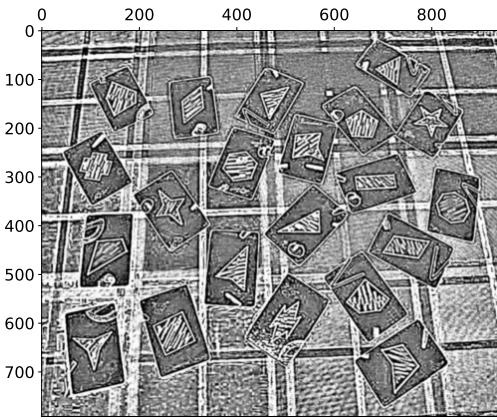
Таким образом, на данном этапе имеем алгоритм, выделяющий из входного изображения маску фигур на карточках, довольно точную и качественную.

Распознавание фигур

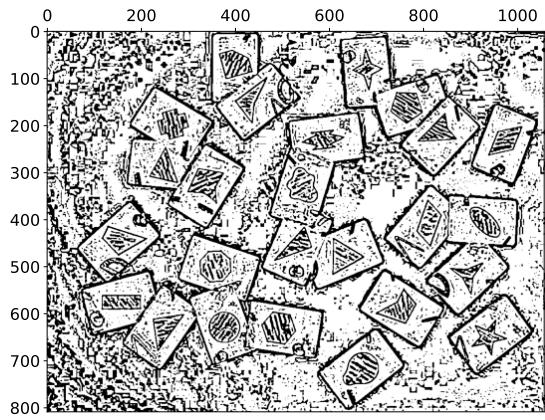
Имея бинарное изображение - маску фигур - воспользуемся реализацией алгоритма Дугласа-Пекера для аппроксимации многоугольников на изображении. В ходе подбора точности приближения было замечено, что чаще всего гладкие фигуры при «средней» точности, при которой почти все многоугольники определяются верно, аппроксимируются 8-угольниками (когда не 8-угольники, то число углов больше, но для этого надо достаточно увеличить точ-



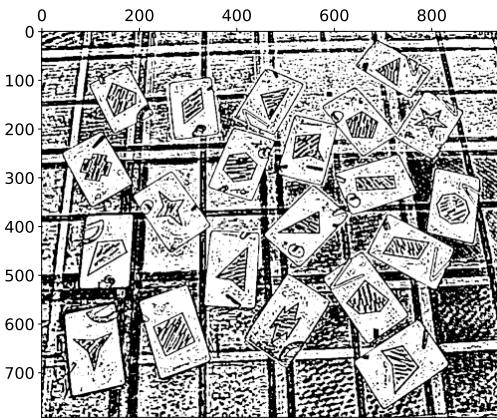
(a) IMG_4.jpg, усилены границы, размыты шумы



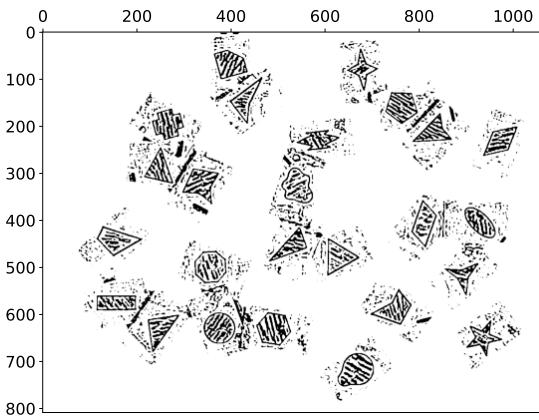
(b) IMG_9.jpg, усилены границы, размыты шумы



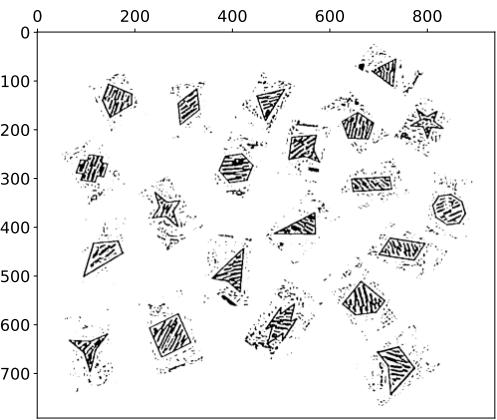
(c) IMG_4.jpg, бинаризация Оцу



(d) IMG_9.jpg, бинаризация Оцу



(e) IMG_4.jpg, после применения маски карточек - контуры фигур и шумы

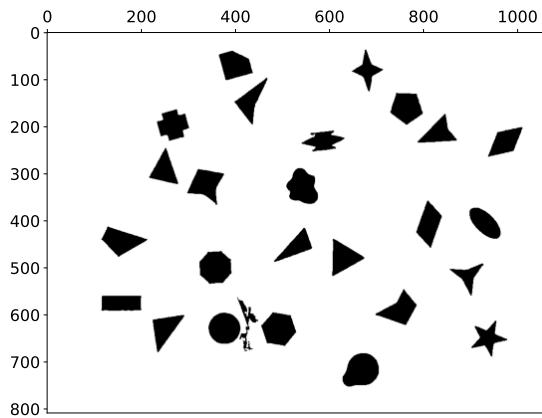


(f) IMG_9.jpg, после применения маски карточек - контуры фигур и шумы

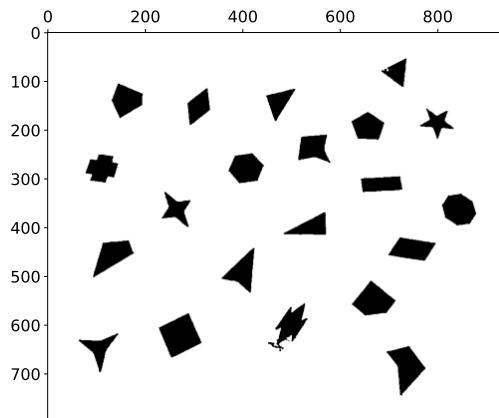
Рис. 10: Получение контуров фигур из эквализированного результата

ность алгоритма). Используя это наблюдение и упомянутый в начале принцип «разделяй и властвуй» было принято разделить все фигуры на изображении на 4 группы:

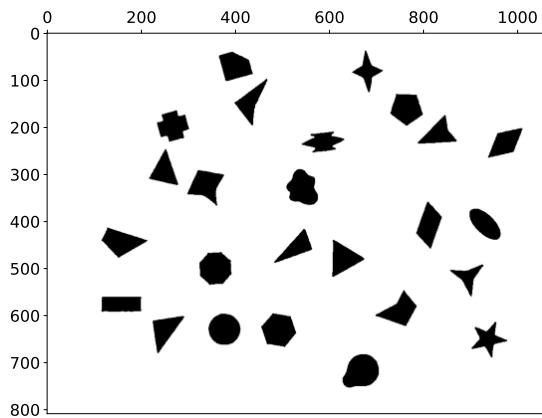
1. Многоугольники с числом сторон < 8
2. Многоугольники с числом сторон $= 8$, то есть 8-угольники
3. Многоугольники с числом сторон > 8



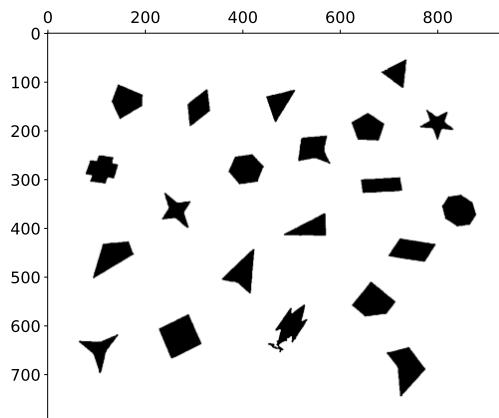
(a) IMG_4.jpg, морфологические операции #1



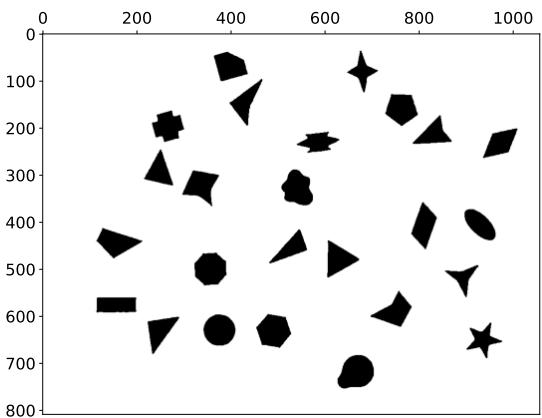
(b) IMG_9.jpg, морфологические операции #1



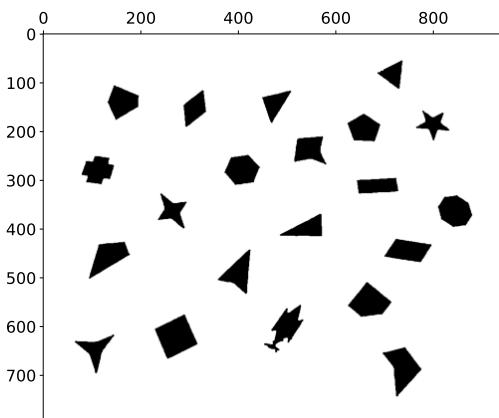
(c) IMG_4.jpg, морфологические операции #2



(d) IMG_9.jpg, морфологические операции #2



(e) IMG_4.jpg, морфологические операции #3

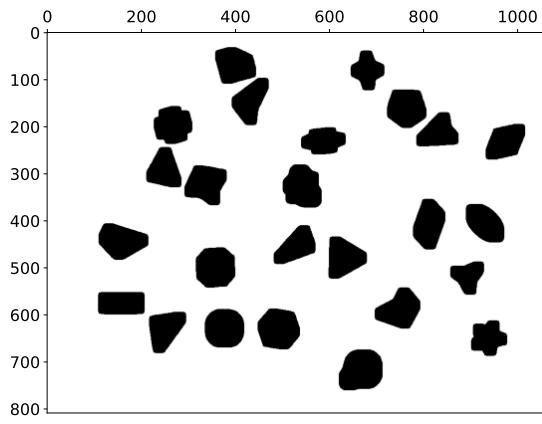


(f) IMG_9.jpg, морфологические операции #3

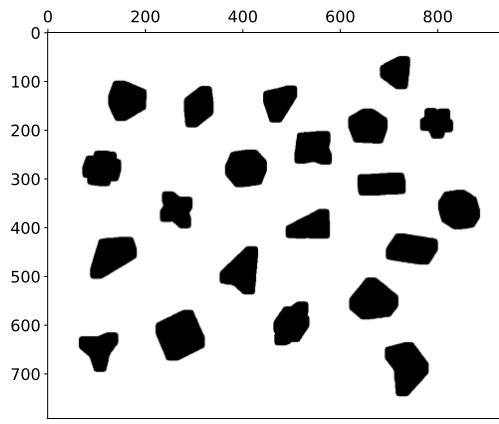
Рис. 11: Получение предфинальной маски фигур

4. Гладкие фигуры

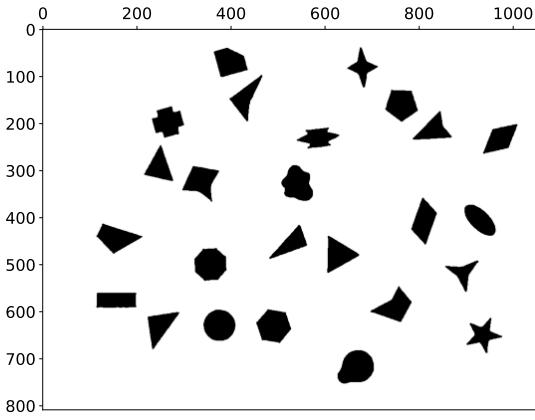
Если взять большую точность алгоритма Дугласа-Пекера, то в небольших зазорах или полостях будут распознаваться углы, а если точность будет маленькой, то небольшие углы не будут замечены. Обойтись одной точностью не получилось, и может не получится в общем случае, даже, если маска очень хорошая, в силу того, что у фигур может быть разный масштаб. Например, в случае большого треугольника и маленького десятиугольника при большой точности (для распознавания десятиугольника) из-за зазоров или из-за небольшой гладкости углов треугольника он может аппроксимироваться четырёхугольником. Таким образом, кажется разумным распознавать отдельно многоугольники с малым и большим числом вершин. А так



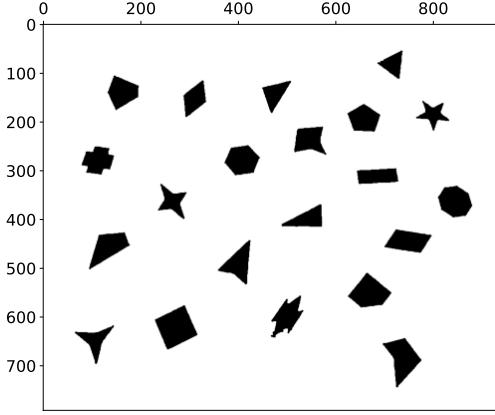
(a) IMG_4.jpg, дополнительная маска, полученная из предфинальной



(b) IMG_9.jpg, дополнительная маска, полученная из предфинальной



(c) IMG_4.jpg, финальная маска



(d) IMG_9.jpg, финальная маска

Рис. 12: Получение финальной маски фигур

как в задаче так же присутствуют гладкие фигуры, которые чаще всего «маскируются» под восьмиугольники, имеет смысл рассмотреть последние отдельно.

Итак, фигуры распознаются следующим образом:

1. Делаем копию маски фигур. На ней делаем закрытие и уточняем её: для устранения возможных дефектов контуров фигур. А именно: сначала возможные выбоины закрываются (так неполучится сделать с фигурами с большим числом вершин, ведь одной из таких закрытых «выбоин» может оказаться угол невыпуклого многоугольника), потом убирается гладкость с углов за счёт уточнения маски. Далее проводится «страховочное» нерезкое маскирование. Находим контуры, **ставим среднюю точность**, аппроксимируем многоугольниками и все фигуры, которые распознаются как многоугольники с числом вершин < 8 , такими и объявляются: контуры этих фигур запоминаются, а в маске фигур место этой фигуры заполняется (прибавление к маске дилатированной заливки по запомненному контуру), чтобы она не «мешала» при распознавании фигур на следующих этапах.
2. Делаем копию маски (в ней уже нет многоугольников с числом углов < 8). На ней делаем небольшое уточнение для более выразительных углов. Находим контуры, **ставим высокую точность** и аппроксимируем многоугольниками - все восьмиугольники распознаются восьмиугольниками. А гладкие фигуры из-за высокой точности аппроксимируются гораздо большими многоугольниками. То есть всё, что аппроксимировалось восьмиугольником, объявляем восьмиугольником, запоминаем контур и удаляем расположенные

знанный восьмиугольник из маски.

3. Делаем копию маски (в ней уже нет многоугольников с числом углов < 8 и восьмиугольников). На ней делаем только нерезкое маскирование (чем больше углов, тем больше мест исказить фигуру, поэтому просто увеличиваем резкость). Находим контуры, **ставим точность чуть ниже среднего** (чтобы у всех гладких фигур наверняка аппроксимировалось не более восьми углов), аппроксимируем многоугольниками, и всё, что распозналось многоугольником с числом вершин > 8 , объявляем ими, запоминаем контуры и удаляем распознанный многоугольник из маски.
4. В оставшейся маске фигур многоугольников нет. Если есть какие-то замкнутые контуры - находим их, это гладкие фигуры.

Выпуклость многоугольников определяем по уже реализованному алгоритму, основываясь на том, что выпуклый многоугольник совпадает со своей выпуклой оболочкой.

На рассмотренных изображениях все фигуры распознались верно и с достаточно точными контурами. На результатах с заполнением фигур цвет означает следующее:

- Многоугольники с числом вершин < 8 залиты **синим** цветом.
- Восьмиугольники залиты **зелёным** цветом.
- Многоугольники с числом вершин > 8 залиты **жёлтым** цветом.
- Гладкие фигуры залиты **бирюзовым** цветом.

Границы фигур выделены **красным** цветом.

Итого, имеем алгоритм распознавания фигур на карточках.

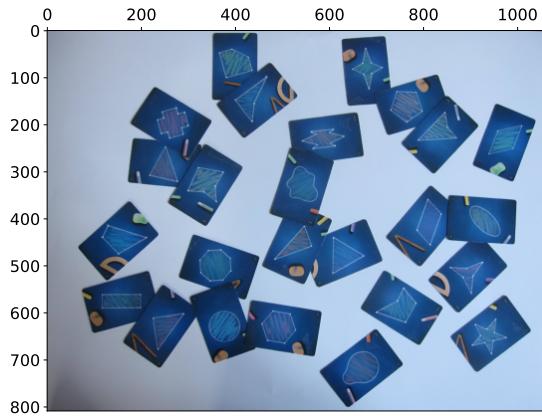
Программная реализация

Программа написана на языке программирования Python в среде Jupyter Notebook с использованием библиотек:

- numpy для векторных и матричных вычислений
- matplotlib для отрисовки графиков
- pillow для ввода и вывода изображений
- os для работы с системными директориями (чтение, запись)
- scikit-image, scipy.ndimage для операций над изображениями
- OpenCV для поиска контуров, аппроксимации и отрисовки границ фигур, для вывода изображений в окнах

Программа находится в Jupyter Notebook, который состоит из импортов, вспомогательных функций, алгоритма решения задачи в нескольких функциях (сегментация карточек, сегментация фигур и распознавание фигур), примера с выводом всех промежуточных данных, экспериментов на всех данных, и, собственно, программы (ячейки после «Program»).

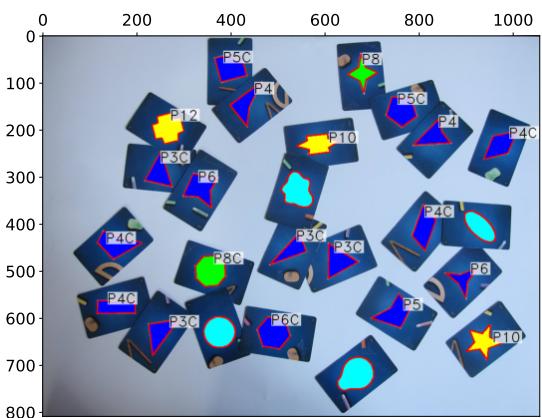
Для взаимодействия с программой нужно запустить последнюю ячейку, вписать путь к входному файлу, то есть обрабатываемому изображению.



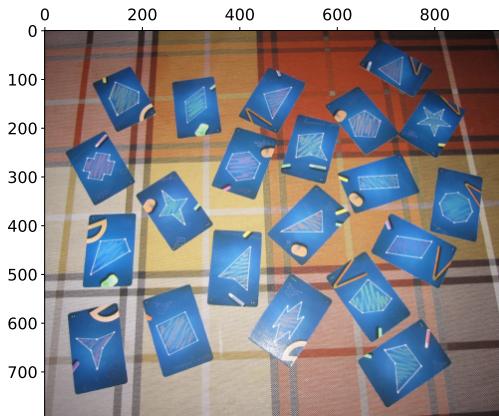
(a) IMG_4.jpg, оригинал



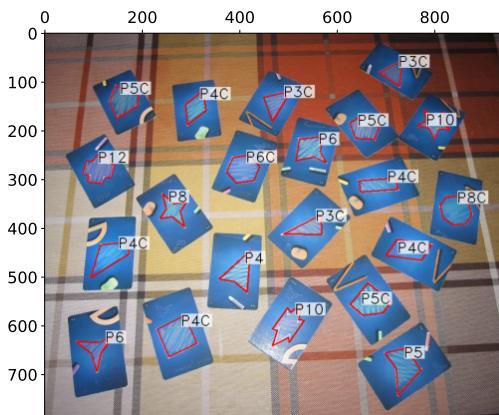
(c) IMG_4.jpg, размеченное изображение



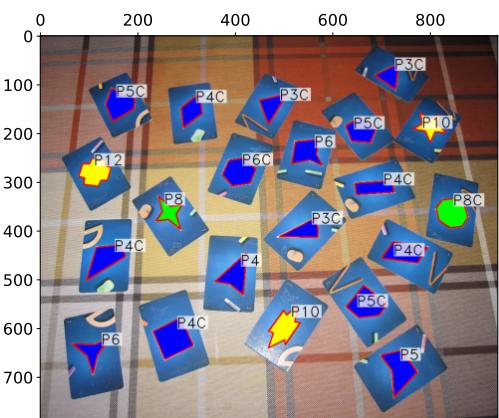
(e) IMG_4.jpg, размеченное изображение, с
заполнением фигур



(b) IMG_9.jpg, оригинал



(d) IMG_9.jpg, размеченное изображение



(f) IMG_9.jpg, размеченное изображение, с
заполнением фигур

Рис. 13: Оригинальное изображение и результат распознавания

Далее появится окно с входным файлом, нужно нажать любую точку. После программа начнёт работать и через несколько секунд появится окно с результатом работы, размеченным изображением.

Перед выводом размеченного изображения в консоль выводится число фигур, оно же число карточек, которые есть на изображении и перечислены все фигуры с типами и координатами центров.

Также перед работой программы предлагается настроить её - настроить перед запуском значения следующих переменных в ячейке:

```
In [*]: # you can set parameters and flags
input_folder = 'input'
vis_folder_ = 'vis'
output_folder = input_folder

chk_dump_vis = False
chk_save_res = True
chk_fill = True
chk_border = True
chk_label = True

# program
name = input()

try:
    process(name, chk_dump_vis=chk_dump_vis, chk_save_res=chk_save_res,
            chk_fill=chk_fill, chk_border=chk_border, chk_label=chk_label)
except:
    print('Error! Something gone bad!')

IMG_4.jpg
```

Рис. 14: Запуск программы и ввод пути к изображению

```
try:
    process(name, chk_dump_vis=chk_dump_vis, chk_save_res=chk_save_res,
            chk_fill=chk_fill, chk_border=chk_border, chk_label=chk_label)
except:
    print('Error! Something gone bad!')

IMG_4.jpg
Figures quantity: 25
1. P3C; center coords: (249, 631)
2. P6C; center coords: (490, 630)
3. P4C; center coords: (157, 575)
4. P5; center coords: (746, 584)
5. P6; center coords: (890, 518)
6. P3C; center coords: (627, 478)
7. P3C; center coords: (529, 453)
8. P4C; center coords: (152, 442)
9. P4C; center coords: (809, 408)
10. P6; center coords: (339, 328)
11. P3C; center coords: (248, 291)
12. P4C; center coords: (970, 232)
13. P4; center coords: (835, 210)
14. P5C; center coords: (761, 160)
15. P4; center coords: (433, 148)
16. P5C; center coords: (399, 66)
17. P8C; center coords: (356, 499)
18. P8; center coords: (678, 81)
19. P10; center coords: (933, 649)
20. P10; center coords: (584, 230)
21. P12; center coords: (266, 197)
22. smooth; center coords: (661, 717)
23. smooth; center coords: (374, 626)
24. smooth; center coords: (928, 405)
25. smooth; center coords: (539, 328)
```

Рис. 15: Вывод программы

- `input_folder` - директория, в которой находится изображение, имя которого введено
- `vis_folder` - директория, куда (при необходимой настройке) сохранять промежуточные результаты работы программы
- `output_folder` - директория, куда (при необходимой настройке) сохранять результаты работы программы (размеченное изображение и текстовое описание фигур на изображении)
- `chk_dump_vis` - флаг, отвечающий за вывод и сохранение промежуточных результатов работы программы

- `chk_save_res` - флаг, отвечающий за сохранение результатов работы программы
- `chk_fill` - флаг, отвечающий за заполнение областей внутри фигур на размеченном изображении
- `chk_border` - флаг, отвечающий за рисование границ фигур на размеченном изображении
- `chk_label` - флаг, отвечающий за текстовую разметку фигур на изображении

Эксперименты

Предложенный алгоритм распознавания фигур на карточках был протестирован на всех предлагаемых образцах. Результаты на [рис. 16](#).

По результатам экспериментов на всех входных изображениях программа верно и точно (с качественными границами) распознала фигуры.

Заключение

Итого, получена программа для распознавания фигур на изображениях карточек игры «Геометрика». Данный подход написания программы соответствует обучению без учителя: предоставленные в качестве образцов изображения были использованы только для отладки и подбора параметров. И несмотря на недостаток в виде возможной необходимости заново подбирать гиперпараметры на новых данных, у алгоритма есть существенное преимущество - отсутствие процедуры обучения, а значит и простое и быстрое использование.

Источники

1. [Обработка и распознавание изображений \(курс лекций, Л.М. Местецкий\)](#)
2. [Документация библиотеки scikit-image](#)
3. [Документация библиотеки OpenCV](#)



(a) IMG_1.jpg, размеченное изображение



(b) IMG_2.jpg, размеченное изображение



(c) IMG_4.jpg, размеченное изображение



(d) IMG_6.jpg, размеченное изображение



(e) IMG_7.jpg, размеченное изображение



(f) IMG_9.jpg, размеченное изображение



(g) IMG_10.jpg, размеченное изображение

Рис. 16: Результаты экспериментов