

**Московский государственный университет
имени М. В. Ломоносова**



**Факультет Вычислительной Математики и Кибернетики
Кафедра Математических Методов Прогнозирования**

**ЛАБОРАТОРНАЯ РАБОТА №1
ПО КУРСУ «ОБРАБОТКА И РАСПОЗНАВАНИЕ ИЗОБРАЖЕНИЙ»**

**РАСПОЗНАВАНИЕ ФИШЕК ТРИМИНО И ИХ
МАРКИРОВКИ ПО ФОТОГРАФИЯМ**

Выполнил студент 3 курса
317 группы:

Иванов Сергей Максимович

Москва, 2010

Оглавление

1. Задача.....	3
1.1. Постановка задачи.....	3
1.2. Входные данные	3
1.3. Используемые алгоритмы	4
1.4. Программная реализация	4
1.5. Эксперименты.....	5
2. Определение центров фишек	9
2.1. Базовый алгоритм.....	9
2.2. Недостатки базового алгоритма	12
2.3. Ансамблирование	13
2.4. Объединение результатов.....	15
3. Определение маркировки	17
3.1. Isolation Forest.....	17
3.2. Применение Isolation Forest	18
3.3. Ансамблирование	19
3.4. Объединение результатов.....	20
4. Выводы	21

1. Задача

1.1. Постановка задачи

По приведённой фотографии в формате BMP24 требуется определить:

- сколько фишек игрового набора тримино содержится на фотографии
- для каждой фишки определить:
 - положение центра фишки на фотографии
 - маркировку фишки

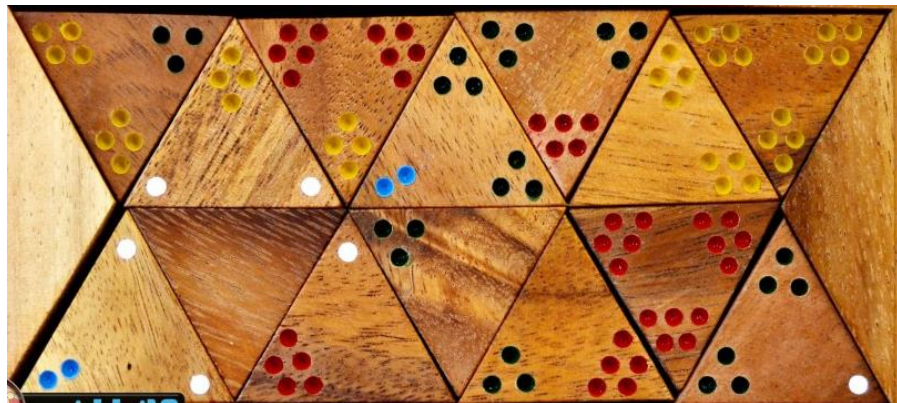


Рисунок 1

Примеры фишек игрового набора тримино приведены на рисунке 1. Маркировкой фишек называется набор из трёх множеств точек (маркеров), расположенных по углам треугольников. В одном углу располагается от 0 до 5 маркеров одного цвета.

1.2. Входные данные

Пример входного изображения приведён на рисунке 2. Считается, что положение центра триминошки программа определяет верно, если отклонение от истинного значения составляет не более 60 пикселей.

По предоставленному набору тестовых данных можно определить также следующие свойства входных изображений:

- Входными изображениями являются фотографии триминошек, разложенных на неоднородном пёстром фоне с посторонними объектами.
- Все триминошки на фотографии имеют примерно одинаковый размер. Также все триминошки имеют форму примерно равностороннего треугольника.



Рисунок 2

- Все маркеры на триминошках имеют примерно одинаковый размер и примерно образуют на фотографиях круги.

- Каждому количеству маркеров в одном углу соответствует своему определённому цвету:

0	-	2	зелёный	4	синий
1	белый	3	жёлтый	5	красный

1.3. Используемые алгоритмы

- **Дилатация и эрозия с единичным ядром**

При применении операции *дилатации*, для текущего пикселя рассматривается окрестность, соответствующая размеру ядра. Из яркостей пикселей окрестности выбирается максимум и объявляется результатом операции в текущем пикселе. Применение операции *эрозии* аналогично, но среди пикселей окрестности берётся минимум.

- **Бинаризация по порогу**

Применяется для чёрно-белого изображения. Пикселям, чья яркость превышает некоторый порог *threshold*, присваивается наибольшая яркость (255), остальным – 0.

- **Разбиение бинарного изображения на компоненты связности**

Применяется для бинарного изображения, то есть изображения только с белыми и чёрными пикселями. Формально можно представить изображения в виде графа с вершинами в белых пикселях, рёбра которого соединяют соседние по вертикали или горизонтали пиксели. Процедура возвращает множество компонент связности исходного изображения. Каждая компонента связности также называется *контуром*.

- **Вычисление площади контура**

Возвращает площадь контура, то есть количество пикселей, изолированных точками контура.

- **Медианный фильтр**

Значение пикселя полагается равным медиане значений пикселей в пределах размера ядра. Этот алгоритм помогает избавиться от мелких шумов и сгладить изображение.

- **Поиск минимального круга, целиком содержащего контур**

Задачей алгоритма является нахождение точки, максимальное расстояние от которого до точек контура минимально.


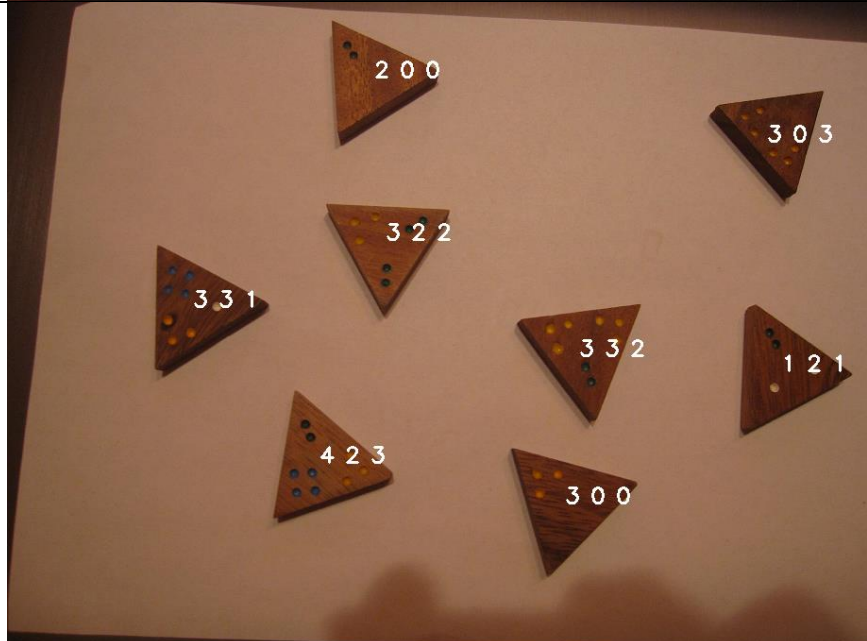
1.4. Программная реализация


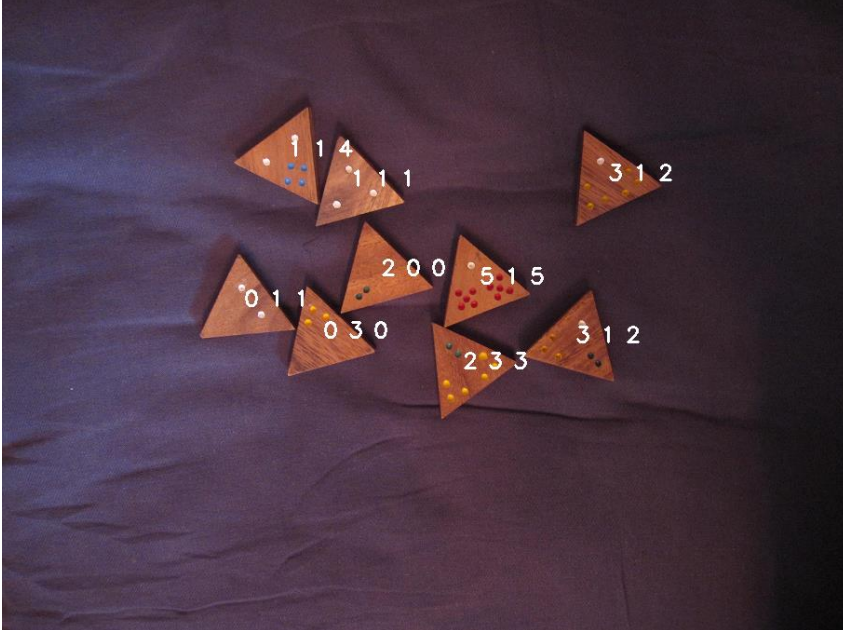
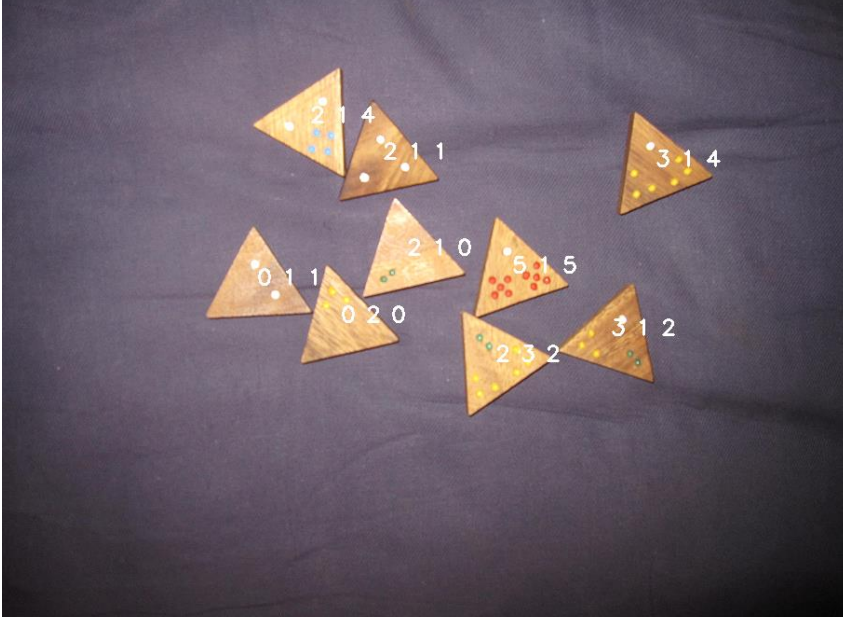
Программа реализована на языке Python 3 в среде Jupyter Notebook. Реализации алгоритмов, перечисленных в разделе 1.3, а также базовые процедуры обработки изображений, взяты из библиотеки OpenCV. Алгоритм изолирующего леса (Isolation Forest) взят из библиотеки sklearn. Этапы работы


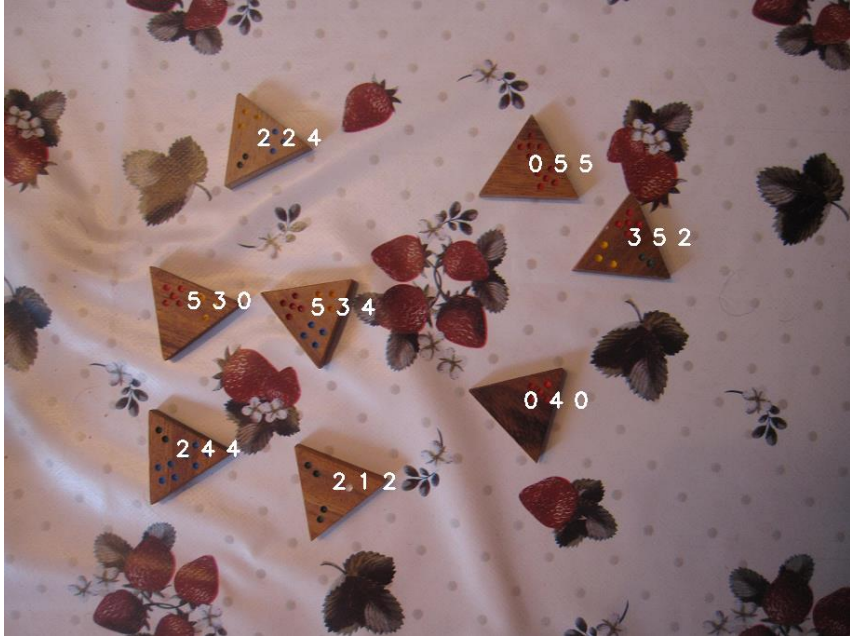

программы соответствуют подразделам 2-3, в которых описан метод решения задачи.

1.5. Эксперименты

В таблице приведён результат работы программы на предоставленных входных изображениях. Точность обнаружения триминошек на них 100%-ая (обнаруживаются все триминошки, лишние не обнаруживаются), поэтому в таблице приведены только результаты распознавания маркеров. Ошибкой I рода считается ложное обнаружение маркера, ошибкой II рода – пропуск маркера.

	Ошибки I рода	Ошибки II рода
	1	0
	1	0

	0	3
	0	1
	3	2

	0	0
	0	2
	0	2

	1	1
--	---	---

На сложности Expert точность алгоритма на тестовых данных составила 98%. В силу стохастичности алгоритма, результат при разных запусках может радикально отличаться, в таблице приведены результаты на одном запуске.

2. Определение центров фишек

2.1. Базовый алгоритм

Рассмотрим следующий алгоритм нахождения равносторонних треугольников на чёрно-белой фотографии (рис. 3а).

Применим к изображению операцию дилатации несколько раз с ядром небольшого квадратного размера (рис. 3б). Размер ядра (`kernel_size`) и количество итераций (`iterations`) являются параметрами алгоритма. Из полученного изображения вычтем оригинал.

Заметим, что полученное изображение (рис. 3в) очень похоже на карту границ, то есть яркость пикселя соответствует степени его граничности. Интуиция такого результата заключается в следующем: для неграничных точек значения локального максимума будет достаточно близко к значению в текущем пикселе, поэтому для них результат операции будет близок к нулю. Для граничных же пикселей это неверно, поэтому в них результат процедуры получится выше.

Бинаризуем полученное изображение по порогу `threshold`, значение которого также является параметром алгоритма (рис. 3г).



Рисунок 3а) оригинальное чёрно-белое изображение

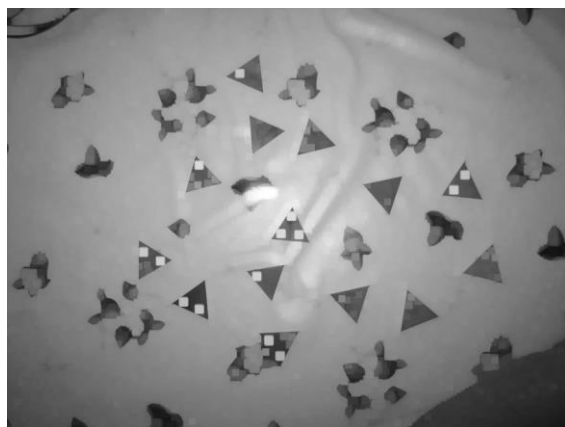


Рисунок 3б) после применения дилатации, `kernel_size = 3`, `iterations = 5`

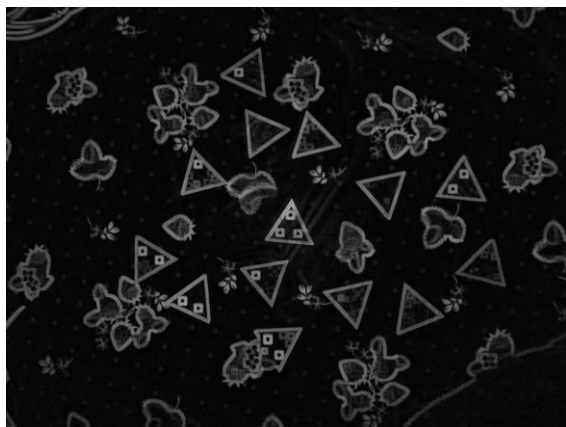


Рисунок 3в) разница 3б и 3а

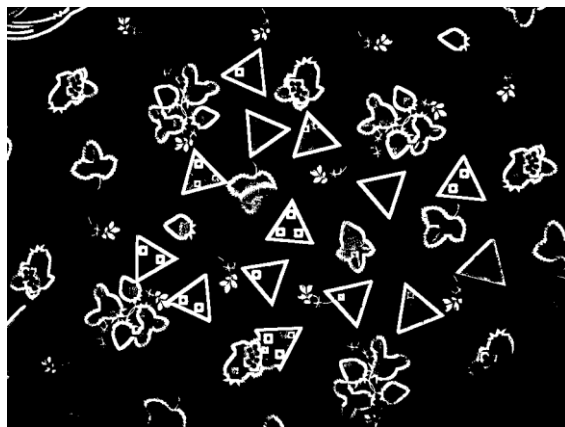


Рисунок 3г) бинаризация по порогу 60

Можно провести симметричные преобразования, используя эрозию, а не дилатацию. Аналогично строится изображение после применения эрозии (рис. 4б), оно вычитается из оригинала (рис. 4в), и к результату применяется бинаризация (рис. 4г). Несмотря на похожесть, итоговое изображение отличается, так как края выделились с более яркой, а с менее яркой стороны. Это означает, что морфология изображения в целом может оказаться другой.

Обозначим *mode* ещё одним параметром алгоритма, который принимает два значения. В одном случае дальнейшая работа ведётся с изображением, полученным через операцию дилатации, в другом – через эрозию.



Рисунок 4а) оригинальное чёрно-белое изображение

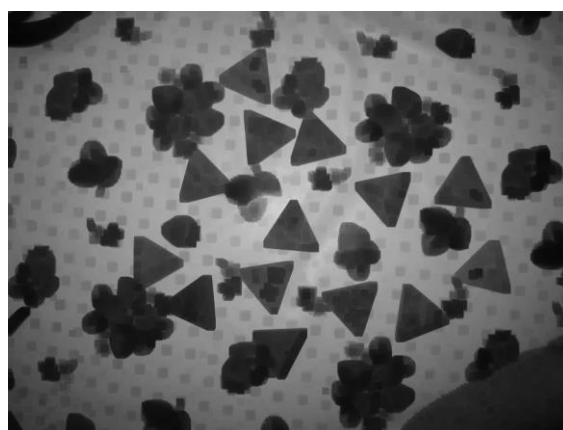


Рисунок 4б) после применения эрозии, *kernel_size* = 3, *iterations* = 5

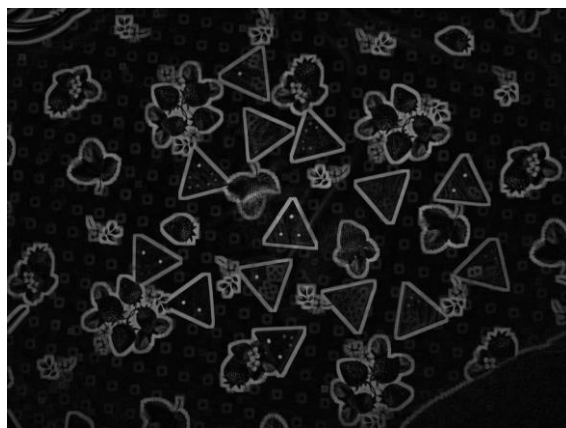


Рисунок 4в) разница 4а и 4б

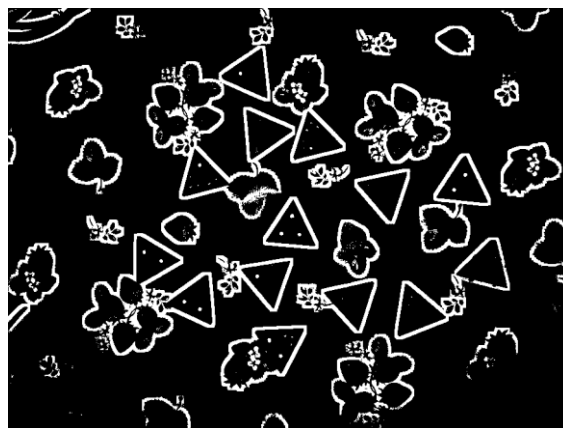


Рисунок 4г) бинаризация по порогу 60

В полученном бинарном изображении выделим компоненты связности. Поставим целью выделить все явно выделившиеся равносторонние треугольники на изображении. Под явно выделившимися подразумевается, что такие треугольники не объединены в одну компоненту связности с другими фигурами, а также, что они образуют замкнутые фигуры (рис. 5а).



Рисунок 5а



Рисунок 5б



Рисунок 5в



Рисунок 5г



Рисунок 5д

На рисунках 5а – 5д приведены различные примеры контуров. Поставим задачу выделения контуров наподобие 5а и игнорирования контуров наподобие 5в – 5д.

Сначала отсечём контура с внутренней площадью менее 1000 точек. Это достаточно заниженный порог на случай появления фотографий с ещё меньшими триминошками. Таким образом, отсекутся очевидно маленькие контура наподобие 5д.

Для каждого контура найдём четыре крайние точки – это точки с минимальными и максимальными координатами по каждой оси (рис. 6а – 6г).

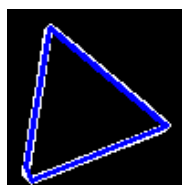


Рисунок 6а



Рисунок 6б



Рисунок 6в



Рисунок 6г

Для идеального треугольника эти четыре вершины будут вершинами треугольника, а это значит, что расстояние между двумя из четырёх точек крайне мало. Посчитаем стороны четырёхугольников и возьмём минимум. Будем считать, что четырёхугольник вырождается в треугольник только если этот минимум не превосходит 10 пикселей. Вершины, между которыми этот минимум был достигнут, усредним (рис. 7а – 7г).

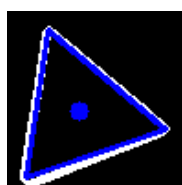


Рисунок 7а



Рисунок 7б



Рисунок 7в



Рисунок 7г

Теперь отбросим неравносторонние треугольники, а также треугольники явно слишком маленького или большого размера. Для последней задачи подсчитаем среднюю длину сторон (*mean*), для первой – дисперсию (*variance*). Отсечение по порогам $mean > 60$, $mean < 200$, $variance / mean < 0.6$ позволяет отсечь контуры 6б и 6в. Данные пороги взяты с запасом, для увеличения обобщающей способности алгоритма, но могут также считаться параметрами алгоритма.



Рисунок 8. Пример работы базового алгоритма

2.2. Недостатки базового алгоритма

На рисунке 7б хорошо видно, что такой алгоритм отбрасывает те треугольники, «шум» на которых нарушает правильность треугольника на граничных точках. Это означает, что главной проблемой такого алгоритма является низкая полнота.

Не менее важной проблемой является большой набор параметров с широким диапазоном потенциально оптимальных значений. Если пороги для финального отбора, `kernel_size` и `iterations` (параметры операции дилатации) можно как-то зафиксировать, то вот порог бинаризации для каждого треугольника оптимален по-своему из-за различного шума и его связанности с посторонними фоновыми фигурами, как на рис. 9.

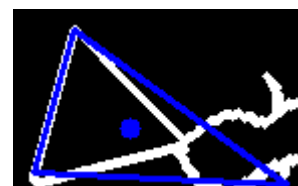


Рисунок 9

Также важно, что финальные пороги отсечения по размеру треугольника приходится выбирать с большим запасом, так как на предоставленных тестовых изображениях разброс длин сторон был достаточно большим. Это означает, что очевидно неправильные по размеру треугольники, как на рис. 7г, остаются неотфильтрованными.

2.3. Ансамблирование

Для решения основных проблем воспользуемся тем, что FPR базового алгоритма крайне низок - алгоритм очень редко выбирает неправильные треугольники. Будем перебирать возможные параметры: $mode \in \{0, 1\}$; $threshold \in [5, 250]$ с шагом 5. Для каждого набора параметров запустим базовый алгоритм и объединим полученные результаты. Пример результата работы ансамбля можно увидеть на рис.10.



Рисунок 10. Объединение результатов многих базовых алгоритмов

Заметим, что помимо вышеуказанных параметров базового алгоритма важно и то, на какой оригинальной картинке он запускается. До этого



Рисунок 11а) результат на синем канале



Рисунок 11б) результат на зелёном канале



Рисунок 11в) результат на красном канале



Рисунок 11г) объединение результатов

рассматривался только grayscale-изображение, однако ничто не мешает запустить его на красном, зелёном или синем канале (рис. 11а – 11в). Итог работы алгоритма приведён на рис. 11г

Видно, что один из треугольников так и не был выделен. Соответствующий треугольник соприкасается тёмной стороной с чёрным фоновым объектом, из-за чего граница размыта на всех четырёх одноканальных изображениях. Это наводит на мысль придумать ещё какой-то канал для подобных ситуаций.

Идея этого канала заключается в том, чтобы построить канал, похожий на канал коричневого цвета. В качестве примера коричневого цвета взят (110, 70, 50) в RGB-координатах. Для этого необходимо задать некоторую метрику расстояния между произвольным цветом (*pixel*) и данным (*brown*).

$$d(pixel, brown) = \max_{c \in \{r, g, b\}} \frac{(pixel_c - brown_c)^2}{norm_c^2}$$

$$norm_c = \max_{p \in [0, 255]} (p - brown_c)$$

Интуиция метрики заключается в желании придумать метрику расстояния между любыми двумя цветами в координатах RGB, штрафующую за высокие отклонения по одной из координат (поэтому выбрана метрика Минковского для $p = \infty$) с нормировкой диапазона возможных расстояний для каждого канала.

Получившийся «коричневый канал» показан на рисунке 13



Рисунок 13



Рисунок 14) Итоговый результат работы ансамбля

2.4. Объединение результатов

На этом этапе в множестве треугольников требуется найти те, которые обрамляют одну и ту же триминошку, и усреднить их ответы. Дополнительно можно попытаться как-то отфильтровать лишние треугольники.



Рисунок 15) Результат после дополнительной фильтрации

В базовых алгоритмах диапазон, по которому отсекались слишком маленькие или большие треугольники, был очень широким в силу отсутствия априорной информации о размерах триминошек. В предположении низкого FPR базовых алгоритмов, можно достаточно уверенно положить, что медиана всех сторон треугольников – очень хорошая оценка настоящего размера. Это позволяет сразу задать достаточно узкий порог (отличие от оценки разрешается не более, чем на 15%), который отсекает как некорректные треугольники, так и некоторые плохие приближения правильных ответов (рис. 15).

Для оставшихся треугольников остаётся задать только критерий того, что два треугольника обрамляют одну и ту же триминошку, и способ усреднения ответов. Критерием объявляется то, что центр одного из треугольников лежит внутри другого. Усреднение же происходит взвешенным усреднением соответствующих точек (соответствие точек дополнительно вычислять не требуется, так как каждой точке треугольника соответствует, является ли она верхней, нижней, правой или левой по построению). В качестве веса ответов треугольника T взята величина $\frac{1}{T.variance}$, где $T.variance$ – дисперсия длин сторон треугольника T .

Итоговый результат работы показан на рисунке 16.



Рисунок 16) Итоговый результат

3. Определение маркировки

3.1. Isolation Forest

Изолирующий Лес (Isolation Forest) – алгоритм машинного обучения без учителя. Задача алгоритма – отделить в наборе объектов, заданных в некотором признаковом пространстве, аномалии. Аномалиями здесь полагаются объекты, выпадающие из основного распределения; это могут быть как объекты с низкой апостериорной плотностью, так и объекты из кластеров малых размеров. Результатом работы алгоритма является $a_i \in [-1, 1]$ для каждого объекта – степень аномальности (anomaly score), чем ближе a_i к -1, тем вероятнее данный объект является аномалией.

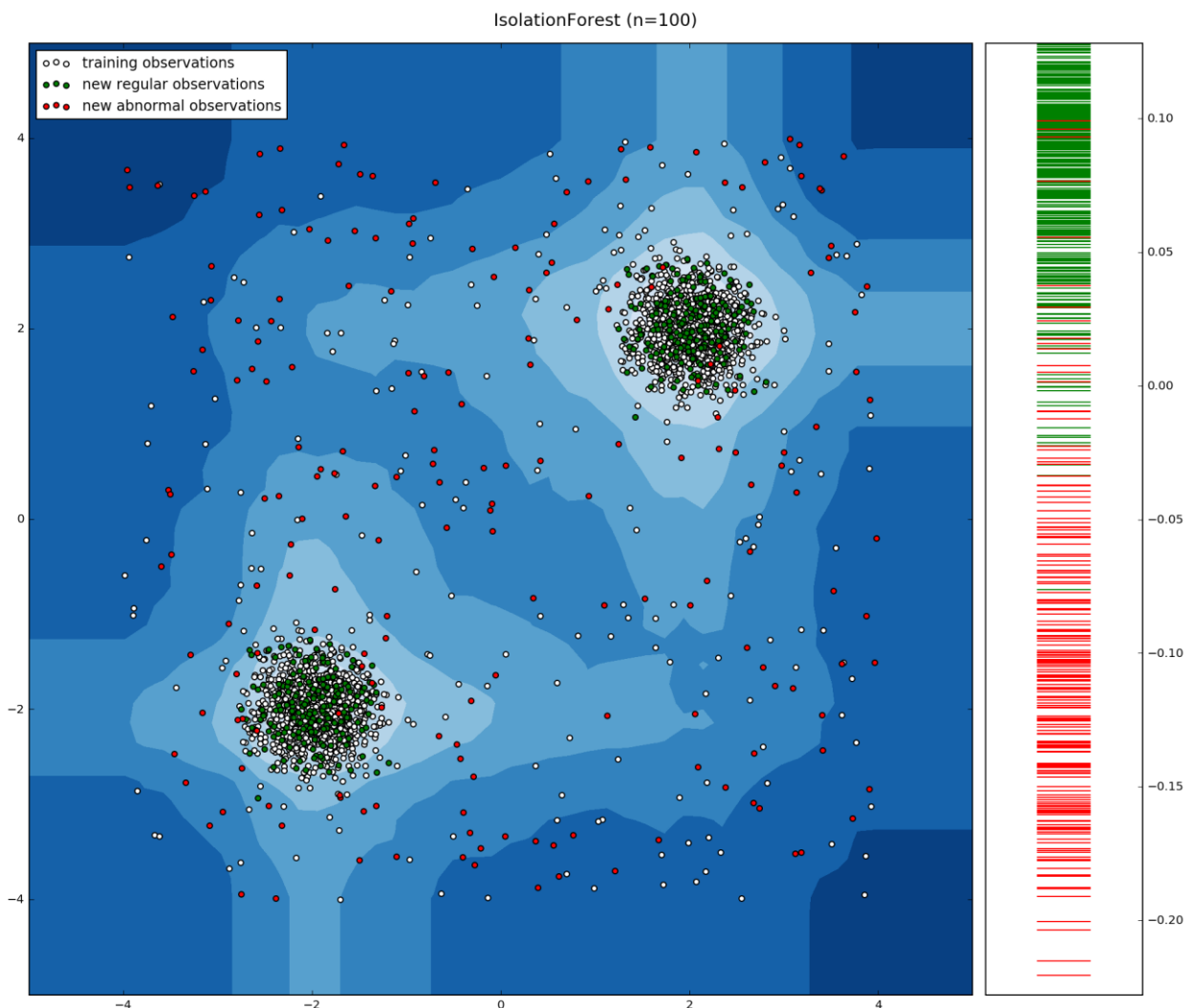


Рисунок 17) Пример работы Isolation Forest. Белые – точки обучения, среди которых как объекты из двух основных кластеров, так и шум. Зелёные и красные точки – тестовая выборка, зелёные точки сэмплированы из того же распределения, что и два основных кластера, красные – шум. Для каждого объекта тестовой выборки справа соответствующим цветом нарисовано его anomaly_score

Принцип работы алгоритма заключается в усреднении аналогичных результатов нескольких (обычно, 100) изолирующих деревьев (Isolation Tree).

Каждое такое дерево является полностью случайным решающим деревом. В каждом узле для предиката выбирается совершенно случайный признак и совершенно случайное его значение из равномерного распределения. Дерево строится целиком, пока в каждом узле не окажется по одному объекту. Степень аномальности объекта при этом объявляется тем выше, чем ближе объект находится к корню дерева (чем меньше глубина листа, в котором он находится). Алгоритм строит такие деревья по обучающей выборке и способен выдавать прогноз в том числе и для новых точек, вычисляя, в каком листе дерева окажется новый объект.

3.2. Применение Isolation Forest

Рассматривая только пиксели одной триминошки, можно заметить, что значения каналов RGB на ней в области точек сильно отличаются от значений коричневых пикселей. Это означает, что изолирующий лес должен хорошо отделить кластер коричневых пикселей триминошки от меньших кластеров цветных (например, красных) пикселей.

Для каждого найденного на первом этапе треугольника переберём все точки внутри него. Каждый такой пиксель будем рассматривать как объект, а значения каналов RGB – как признаки. Обучим алгоритм на данном множестве объектов.

В принципе, достаточно было бы и применить алгоритм к точкам внутри треугольника, однако в силу не идеального выделения границ на первом этапе, точки могут оказываться на краю и частично границами обрезаться, что будет мешать последующему выделению контуров. Поэтому для получения предсказаний алгоритму подаётся прямоугольник, содержащий выделенный треугольник (рис. 18а). На выходе алгоритм выдаёт для каждого пикселя *anomaly score*, вещественное число, близкое к 1 для коричневых точек, и к -1 – для реже встречающихся. Приведём результат к диапазону $[0, 255]$ и инвертируем (рис. 18б). Наблюдается небольшой шум – чуть выцветшие пиксели дерева алгоритм также склонен считать возможными аномалиями. Для избавления от этого эффекта к результату применяется медианный фильтр с ядром размера 3 (рис. 18в).



Рисунок 18а

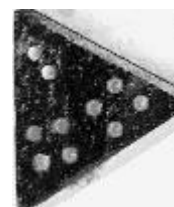


Рисунок 18б



Рисунок 18в

Заметим, что алгоритм в целом отделяет цветные маркеры от поверхности триминошки и фона. Проведём бинаризацию с некоторым порогом (рис. 19).

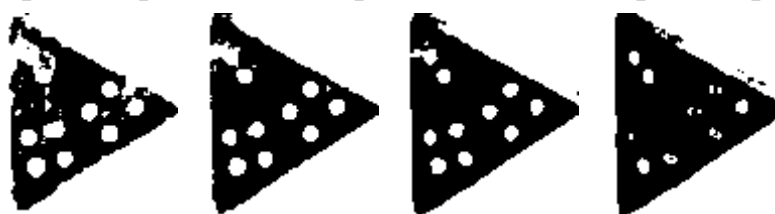


Рисунок 19) Бинаризация по порогам 60; 90; 120; 145

Можно заметить, что при разных порогах явно выделяются разные маркеры. Также хорошие пороги отличаются для разных триминошек, а их подбор неочевиден. Поэтому применим стратегию, похожую на использованный при детектировании триминошек – для каждого порога из некоторого набора (от 60 до 150 с шагом 5) переберём все связанные компоненты и запомним похожие на круги.

Контур можно считать похожим на круг, если его площадь близка к площади минимального круга, целиком содержащего контур. Однако, поскольку маркеры достаточно маленькие, это приближение достаточно условное, и потому отсекаются только те контура, отношение площади которых к площади описанного круга меньше 0.5. Также отбрасываются слишком большие контура (площадью более 100) и маленькие (шумы), чей размер меньше $\frac{side_size^2}{500}$, где *side_size* – медиана сторон всех найденных треугольников. Результат работы алгоритма показан на рисунке 20.

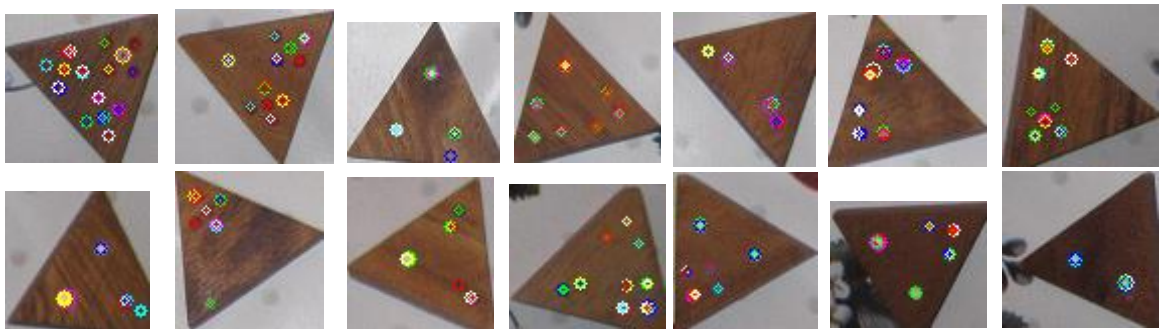


Рисунок 20

3.3. Ансамблирование

В силу того, что изолирующий лес – стохастический алгоритм, а результат зависит от того, насколько хорошо «легли» скоры аномальности, для увеличения точности предлагается запустить его несколько раз и объединить результаты.

В итоговом ансамбле изолирующий лес запускается на всех подмножествах признаков (их всего 7 – {R}, {G}, {B}, {R, G}, {R, B}, {G, B}, {R, G, B}), при этом на полном наборе – трижды. Итого получается 9 алгоритмов. Обосновано это тем, что, например, красные точки, казалось бы, проще определять только по красному каналу, а жёлтые – по {R, G}. В целом, алгоритм хорошо работает и при однократном запуске на всех трёх признаках,



Рисунок 21

однако такой подход позволяет увеличить как точность, так и полноту. Результат работы ансамбля показан на рисунке 21.

3.4. Объединение результатов

В подобном ансамбле, однако, также требуется дополнительная фильтрация для отбрасывания случайно выродившихся в круг шумовых контуров при переборе порогов бинаризации. Поступая аналогично триминошкам, возьмём медиану по площадям всех найденных контуров и будем использовать это значение как оценку «правильного» размера маркеров. Контур, значительно отличающийся от этой оценки (более чем на 60%), отфильтруем. Такая фильтрация отбрасывает заведомо неверные варианты, например, слишком большого размера.

При объединении результатов приходится учитывать, что круги, соответствующие разным маркерам, могут пересекаться (особенно часто это свойственно пятёркам красных маркеров). Поэтому пересечение кругов плохо показало себя критерием того, что два круга искали одну точку. Лучше показывает тот же критерий, что и для треугольников – центр одного из кругов должен находиться внутри другого. При объединении всех таких кругов находится среднее арифметическое их центров и радиусов, и соответствующий круг объявляется окончательным ответом. Результат показан на рисунке 21.

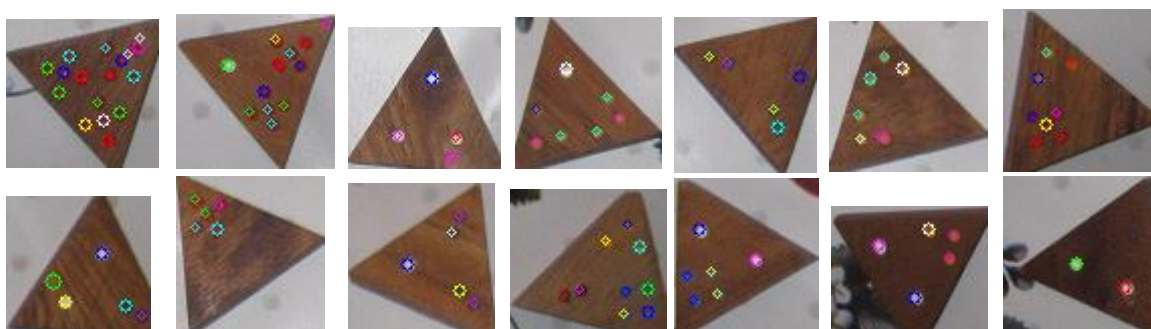


Рисунок 21

Дополнительную фильтрацию от ошибок первого рода можно провести, проводя голосование между 9-ю алгоритмами ансамбля. Хорошим порогом показало требование, чтобы маркер находили больше половины алгоритмов. Окончательный результат показан на рисунке 22. Видно, что 6 ошибок первого рода превратились в 1 ошибку второго рода.

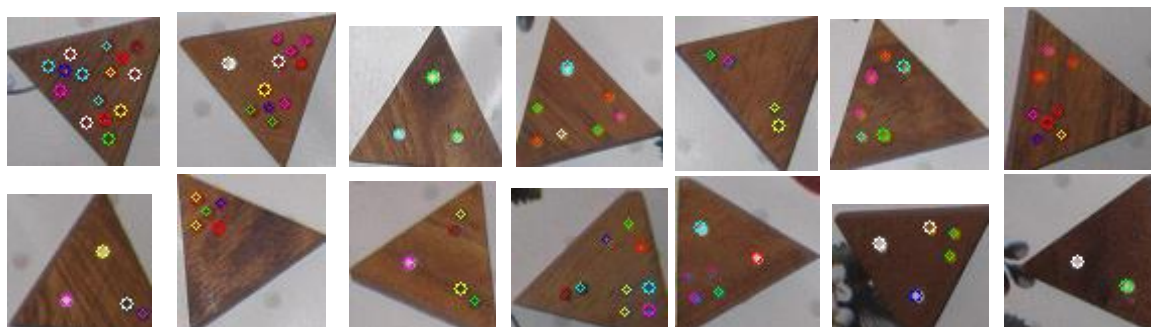


Рисунок 22

4. Выводы

Базовые операции над изображениями, такие как применение простейших пространственных преобразований, позволяют выявить морфологию изображения, найти границы объектов и проанализировать их. В задании хорошо зарекомендовал себя способ объединения результатов работы одного и того же алгоритма с разными параметрами – в первую очередь, это связано с тем, что для разных целевых объектов оптимальные значения этих параметров разные. Свойства медианы позволяют «избавиться от шумов» в том числе и во множестве найденных объектов и отфильтровать лишние варианты.

В задаче распознавания маркеров хорошо показал себя алгоритм поиска аномалий Isolation Forest. При этом вся использованная информация заключалась лишь в значениях каналов RGB пикселей внутри треугольника. Таким образом, методы машинного обучения без учителя работоспособны в задачах распознавания образов на изображениях даже без применения комплексной предварительной обработки.