Отчет по лабораторной работе №1

Алексей Шолохов, 474 группа ФУПМ

6 апреля 2018 г.

1 Постановка задачи

Разработать и реализовать программу для работы с изображений фишек игрового набора Тантрикс, обеспечивающую:

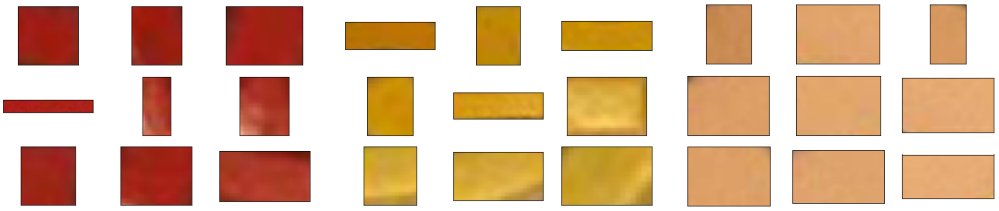
- Ввод и отображение на экране изображений в формате BMP;
- Сегментацию изображений на основе точечных и пространственных преобразований;
- Генерацию признаковых описаний фишек на изображении;
- Классификацию отдельных фишек и их последовательностей.

2 Описание данных

В процессе работы были сформированы два дополнительных набора данных:

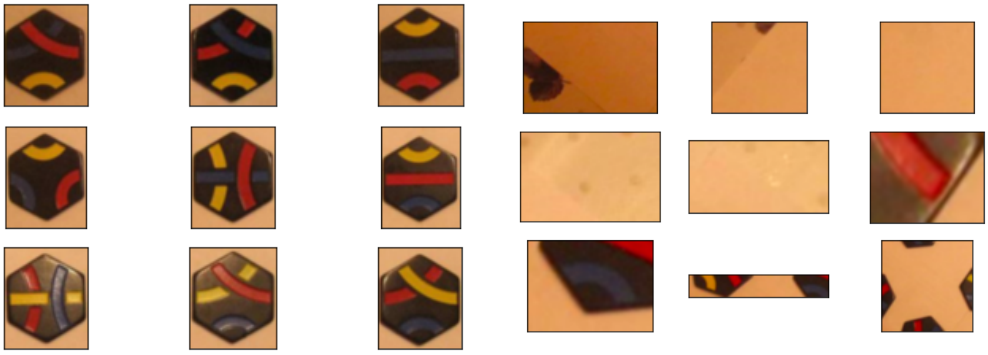
2.1 Набор для определения характеристик цветовых компонент

Видно, что любая картинка из набора имеет пять хорошо заметных цветовых групп: красные, синие и желтые полосы, а также цвет фишки и цвет поверхности, на которой они лежат. Для выделения данных цветовых диапазонов был собран набор данных, состоящий из фрагментов картинок, сгруппированных по цветам. Примеры элементов этого набора можно увидеть на рисунке 1.



(a) Красные линии (b) Желтые линии (c) Поверхность стола

Рис. 1: Различные фрагменты входных картинок



(a) Фишки (b) Не-фишки

Рис. 2: Набор данных для обучения классификатора "фишка – не-фишка"

2.2 Набор фишек и не-фишек

Алгоритм должен иметь встроенный классификатор, который бы отличал фишки от остальных объектов в процессе работы. Для обучения такого классификатора был создан отдельный набор данных формата "двухклассовая классификация": фишки и не-фишки. Примеры объектов из первой и второй группы можно увидеть на рис. 2.

3 Описание метода решения

Структуру решения задачи можно условно поделить на шесть этапов:

1. Выделение цветовых диапазонов для элементов сцены
2. Сегментация изображения
3. Детекция фишек на сегментах, отсев неинформативных сегментов

Элемент	Цветовой диапазон в HSV
Желтая линия	(15-25, 140-255, 150-255)
Красная линия	(0-10, 110-255, 110-255)
Синяя линия	(110-180, 0-100, 25-175)
Цвет стола	(12-15, 110-200, 180-255)

Таблица 1: Таблица цветовых диапазонов для информативных элементов изображений. Видим, что цветовые диапазоны в HSV-формате не пересекаются.

4. Выделение цветовых и геометрических характеристик фишки
5. Классификация фишек по номеру
6. Вывод ответа в требуемом формате

Ниже подробно описан каждый из них:

3.1 Выделение цветовых диапазонов

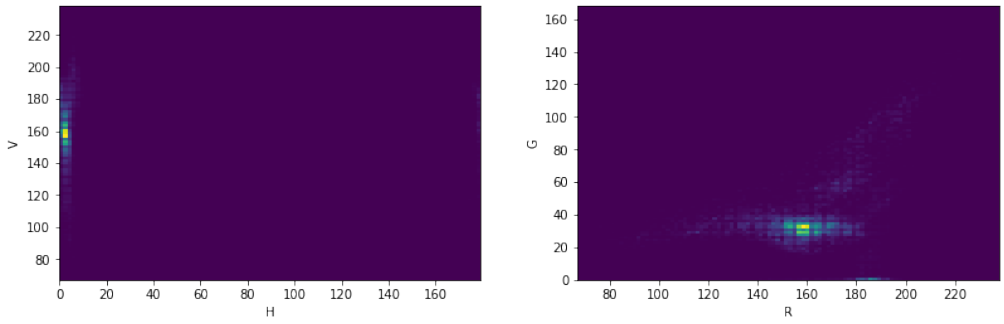
Заметим, что картинки из набора данных получены в неодинаковых условиях: отличается освещение, насыщенность картинки, четкость изображений. Это все делает крайне сложным работу с изображениями в формате RGB – диапазоны цветовых компонент получаются слишком широкими и, порой, перекрывают друг друга: например цвет желтой полосы и цвет фона. В связи с этим было принято решение работать с изображениями в формате HSV, который давал самое компактное представление цветов на гистограммах (пример можно увидеть на рис. 3).

По итогу работы с гистограммами были выделены цветовые диапазоны для элементов сцены в HSV-формате (см. табл. 1). Из таблицы видно, что цветовые диапазоны не пересекаются, что позволяет разделять элементы изображений достаточно уверенно.

3.2 Сегментация

Разбиение изображений на сегменты было реализовано через watershed – алгоритм. Кратко процесс сегментации можно разбить на три этапа:

1. Примерное выделение зон, где находятся фишки, путем удаления фона и бинаризации изображения.



(a) HSV-формат (b) RGB-формат

Рис. 3: Распределение пикселей красной линии по цветовым каналам в двух разных форматах: HSV-формат и RGB-формат. Видно, что HSV-представление имеет гораздо более компактную область, чем RGB.

2. Разделение сцены на три зоны – "точно фишки "точно не фишки"и "зона неуверенности". Это достигается последовательным морфологическим расширением и дилатацией исходной сцены.

3. Применение watershed-алгоритма из библиотеки OpenCV.

Результат можно видеть на рис. 4. Видно, что алгоритм уверенно разделяет на сегменты даже сложные сцены. Ключевое его достоинство (оно же – недостаток), что алгоритм сегментации никак не использует информацию о том, что фишка – шестиугольная. Это, с одной стороны, быстро позволяет обобщить его на фишки произвольной формы, но, с другой стороны, несколько снижает точность сегментирования. Так мы видим, что далеко не все сегменты содержат только лишь фишки, следовательно нужен способ как-то отличать информативные сегменты от неинформативных.

3.3 Детекция фишек на сегментах

Для распознавания фишек на сегментах был обучен классификатор. Процесс его работы можно разделить на три этапа:

1. Препроцессинг фрагмента. Из изображения маской удалялся фон, потом применялось медианное сглаживание с окном 7 с целью удаления шума, оставленного после удаления фона. После этого изображение обрезалось и нормировалось в квадрат 100×100 пикселей.

2. Выделение признаков. К изображению применялись параллельно три маски, каждая из которых выделяла свою линию из изображения. После этого изображение сглаживалось медианным фильтром и из него извлекались контуры всех фигур. Конечное описание полоски состояло из двух цифр: число пикселей и число сегментов. Признаковое описание картинки – это признаковое описание всех ее полос вместе.

3. Обучение классификатора (если набор – обучающий) или его применение к признаковому описанию (если набор – тестовый).

3.4 Классификация фишек

Так как данных было крайне мало, то для классификации фишек использовался простой метрический классификатор формата "1 ближайший сосед": в качестве фишки-паттерна бралась ближайшая по L_2-норме фишка в признаковом пространстве. Несмотря на простоту реализации, данный классификатор практически не допускал ошибок на тестовых данных.

3.5 Выведение ответа в требуемом формате

Данный этап подробно описан в следующем разделе.

4 Описание програмной реализации

В качестве языка программирования был выбран язык Python, в качестве библиотеки для работы с изображениями – OpenCV, в качестве среды разработки и демонстрации работы – среда Jupyter, в качестве формата – Jupyter Notebook. Для облегчения пользования данной программой было реализовано простое API. Базовый (и единственный) объект – AnnotatedCoin обладает методами, приведенными в таблице 2. Анализ картинки происходит в момент создания объекта AnnotatedCoin, поэтому при вызове методов класса вычислений не производится.

5 Эксперименты

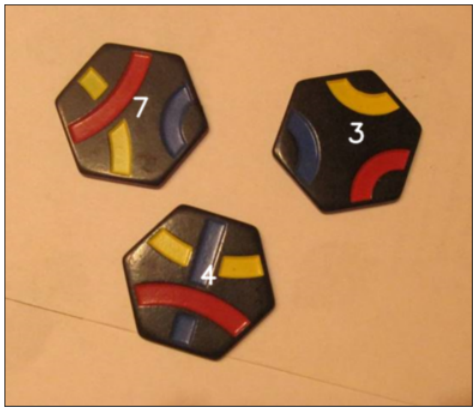
Проанализируем результаты работы программы на предоставленных примерах (рис. 5 .

Рис. 4: Результат применения алгоритма сегментации к изображению

Название	Вход	Описание
Конструктор	uint8 BGR image обучающий датасет	Аннотирует и сохраняет результат
count-coins	нет	Возвращает число фишек на картинке
get-image	нет	Возвращает аннотированную картинку
get-coin-type	нет	Возвращает номер фишки на картинке
get-edges	нет	Возвращает описание ребер фишки

Таблица 2: Методы класса AnnotatedCoin

(a) Пример Group 1



(b) Пример Group 2

(c) Пример Group 4

Рис. 5: Примеры аннотации картинок программой

Видим, что программа работает корректно: все группы по 3 фишки были размечены правильно, несодержательные фрагменты (листочек) на третьей картинки не были приняты за фишку. Примечательно, что при этом не были использованы данные о форме фишки и листика, следовательно, алгоритм достаточно устойчив и универсален. Однако, он делает ошибку путая 6 и 9 фишку. Это происходит потому, что с точки зрения цветовых признаков фишки оказываются практически неразличимы. Следовательно, признаковое пространство стоило бы расширить на другие признаки, например – прямолинейность и криволинейность фрагментов.

6 Вывод

Был успешно реализован функционал, предусматриваемый уровнями «beginner» и «expert». К сильным сторонам реализованной программы стоит отнести независимость сегментации от формы фишки, и, следовательно, легкий перенос программы на фишки другой формы. К слабым сторонам – использование в основном точечных признаков, что в одном случае плохо позволяет алгоритму различить фишку 6 и фишку 9.