Отчёт по выполнению первой лабораторной работы по курсу ОиР изображений

Сидоров Леонид 317 группа

9 апреля 2021 г.

Постановка задачи

Целью работы являются сегментация и классификация изображений. Входом программы является изображение формата bmp. Выходом же программы является число N, характеризующее количество фишек тримино на фотографии, а также N пар координат (X, Y), указывающих положение каждой фишки (её центра), и N троек $(m1, m2, m3)$, отражающих количество точек в углах треугольника — маркировку каждой фишки.

Считается, что положение фишки определилось верно, если отклонение от истинного центра составляет не более 60 пикселов. Примерный размер стороны треугольной фишки равен 130 пикселам.

В тексте задания не регламентируется порядок тройки $(m1, m2, m3)$, поэтому было решено выводить их в порядке убывания.

Было предложено несколько вариантов входных данных. Я выбрал сложность **Expert**. В задании не указывалось никаких ограничений для используемых методов обработки и распознавания изображений.

По итогу выполнения задания должна быть написана программа, принимающая на вход изображения с фишками тримино, описанного ниже, формата и записывающая в файл результаты их сегментации. Воспользуемся введённой выше нотацией и укажем его вид.

На первой строчке должно быть расположено число N, на последующих же N строчках должны присутствовать записи вида $X, Y; m1, m2, m3$. В одном файле могут присутствовать результаты сегментации нескольких изображений, в таком случае они должны отделяться друг от друга пустыми строчками.

Описание данных

Набор изображений **Expert** включает в себя 4 фотографии фишек тримино, разложенных на столе в произвольном порядке (Рис. 1). Фишки сделаны из дерева (цвет неоднороден) и имеют треугольную форму. В каждом углу фишки может находиться от 0 до 5 точек. Каждой группе точек кроме 0 соответствует своя цветовая маркировка: 1 — белый, 2 — зелёный, 3 — жёлтый, 4 — голубой, 5 — красный.

Фоном является скатерть с неоднородным рисунком. Освещение на изображениях *Pict_3_1.bmp* и *Pict_3_2.bmp* отличается от такого на изображениях

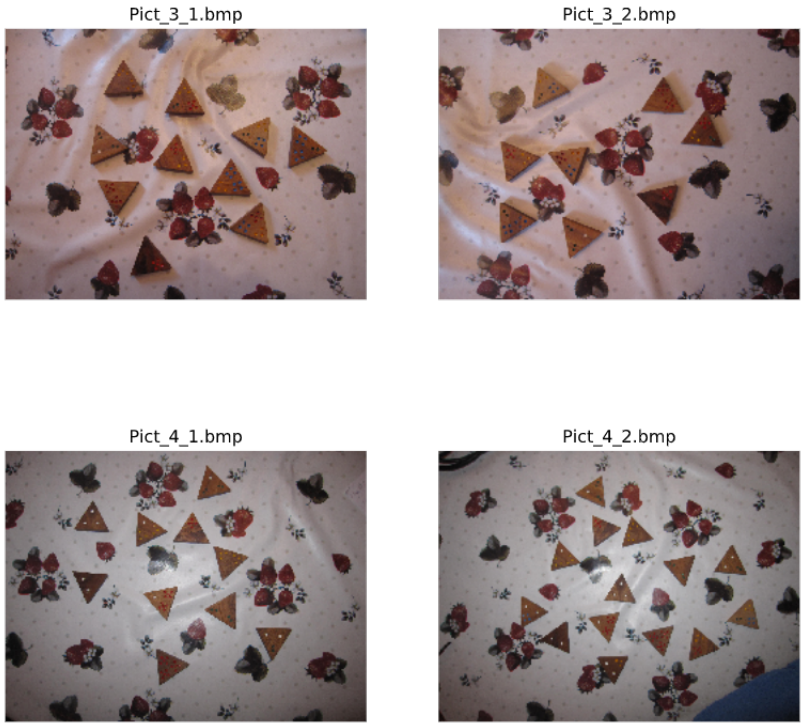

Рис. 1: Набор изображений класса Expert

Pict_4_1.bmp и *Pict_4_2.bmp*. Также стоит отметить, что масштаб снимков из группы 3 немного отличается от масштаба в группе 4.

Размер и разрешение изображений не регламентируются, однако, как уже было сказано выше, сторона фишки должна быть примерно равна 130 пикселам. Изображения из обучающей выборки имеют одинаковый размер (912×684 пикселов), также предполагается, что эти снимки были сделаны на одну камеру.

Описание метода решения

Решение задачи состоит из двух этапов: первый этап заключается в получении сегментационной маски изображения, а второй — в расчёте всех необходимых величин благодаря этой маске.

Сегментация изображения

Шаг 1

Для выделения всех треугольников на фотографии необходимо распознать на ней всё "дерево". Переведём изображение из цветовой схемы **RGB** в цветовую схему **HSV**, она позволяет гораздо эффективнее выделять цветовые диапазоны, ведь главный оттенок там определяется углом от 0 до 360 градусов (также мы можем отдельно задавать яркость и насыщенность цвета).

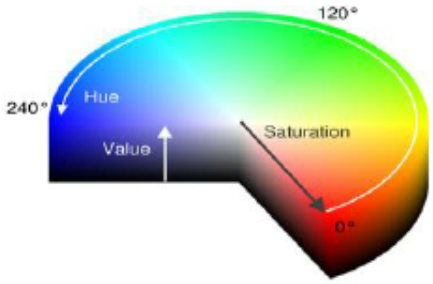

Рис. 2: Цветовая схема HSV

Шаг 2

Такой подход действительно позволил выделить древесный оттенок на фотографии. К полученной бинарной маске применяем операции замыкания и размыкания, это позволит избавиться от мелких объектов фона (узора скатерти) и точек на масках треугольников (маркировки фишек).

Рис. 3: Ядра свёртки

Однако величина порога для применения операции размыкания зависит от масштаба фотографии (величины объектов на ней в пикселах), а потому должна уточняться для каждого изображения отдельно. Без этого уточнения мы рискуем либо удалить маску одного из треугольников, либо оставить на изображении элементы фона. Чтобы оценить масштаб, попробуем сначала подсчитать количество фишек на фотографии.

Шаг 3

Для этого к имеющейся маске применим операции свёртки, ядрами которых являются треугольники со стороной 130 пикселов, повёрнутые на различные углы (Рис. 3). Эти ядра были получены благодаря вырезке и обработке произвольно взятой фишки из тестовой выборки.

Шаг 4

Мы используем 4 ядра свёртки, а потому получаем на выходе 4 матрицы. Чтобы и дальше работать с одним изображением просто сложим их и проведём бинаризацию результата. Выделим на этой маске связные компоненты, их число и будет оценкой на количество фишек. Последовательная обработка изображения представлена на (Рис. 4).

Шаг 5

Теперь оценим масштаб исходной фотографии, сделать это несложно.

$$scale = \frac{step\_2\_pixels}{num\_triag}$$

, где $step\_2\_pixels$ — число ненулевых пикселов на втором шаге, то есть число пикселов (предположительно) занятых фишками, а num_triag — оценка числа фишек.

Используя оценку масштаба, применим к маске, полученной на втором шаге, операцию размыкания повторно, что позволяет окончательно избавиться от шума на сегментационной маске. Результат сегментации также представлен на (Рис. 4).

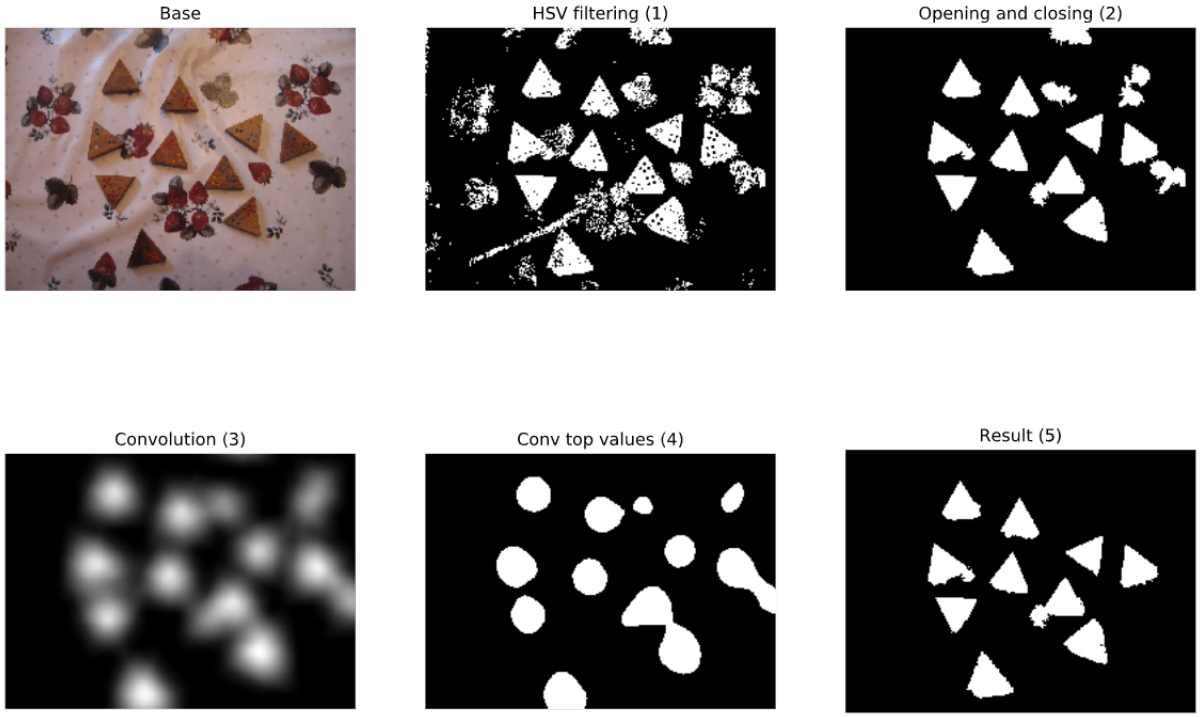

Рис. 4: Последовательность операций для получения числа фишек

Анализ сегментационной маски

Имея сегментационную маску с предыдущего шага, мы можем без труда подсчитать количество фишек и координаты их центров. Количество фишек — это количество компонент связности на этой маске, а координаты их центров — средние координаты этих компонент.

Получение же маркировки является менее тривиальной задачей. Для начала повысим яркость исходного изображения в полтора раза, чтобы цвета стали более выраженными. После этого используем сегментационную маску, чтобы "вырезать"из этого изображения отдельные фишки. На их основе мы получим бинарные маски для каждого из пяти цветов, характеризующих разметку фишки.

Принцип получения масок тот же, что и в предыдущем пункте, при помощи **HSV** представления. Подсчитав количество ненулевых пикселов, принадлежащих к

той или иной маске, мы можем вычислить количественную характеристику того или иного цвета.

На основе этих значений можно определить, сколько и каких маркировок расположено на каждой фишке (проверка порогов с нормировкой на размер изображения).

Если алгоритм в качестве ответа выдал больше трёх значений, то мы просто можем обрезать лишние. Если же значений меньше трёх, то это значит, что как минимум один из углов треугольника не имеет маркировки (соответствует числу 0), и ответ нужно дополнить до трёх чисел нулями.

Описание программой реализации

Программа была написана на языке **Python** с использованием библиотек **OpenCV**, **PIL**, **skimage** и **scipy**.

В папке с программой должны находиться папки *data* и *resources*, в которых должны находиться изображения, подлежащие обработке, и изображение с прообразом ядра свёртки соответственно. В папке *data* могут быть расположены только фотографии для обработки и ничего больше, названия файлов могут быть произвольными. В папке *resources* уже лежит необходимое изображение, на основании которого в шаге 3 предыдущего раздела формируются все необходимые ядра.

По завершении работы программы в её папке должен появиться файл *seg_ output.txt*, который содержит результаты сегментации и классификации каждого изображения, разделённые пустыми строками (формат этого файла описан в первом разделе).

На тестовых данных обработка одного изображения в среднем занимает одну минуту, это вызвано в первую очередь применением громоздкой операции двумерной свёртки с большим ядром (порядка 10000 пикселов). Программа реализует базовую систему логирования, позволяющую понять, какое изображение обрабатывается в данный момент и сколько времени осталось до конца обработки текущего изображения. Эта информация выводится в консоль.

Эксперименты

На (Рис. 5) и (Рис. 6) изображены сегментационные маски и центры фишек из обучающей выборки. Как мы видим, с этой частью задания алгоритм справился на отлично, все маски и координаты соответствуют действительности.

В задаче маркировки фишек алгоритм не достиг абсолютной точности, однако тоже продемонстрировал неплохой результат. В Таблице 1 представлены оценки классификации на основе трёх критериев. Метки $(m1, m2, m3)$ расположены в порядке невозрастания ($m1 \geq m2 \geq m3$). Первая метрика **exact match** возвращает 1 при полном совпадении троек. Вторая метрика **ordinal match** возвращает долю поэлементных совпадений в тройках с учётом порядка. Третья метрика **set match** возвращает долю поэлементных совпадений в тройках без учёта порядка.

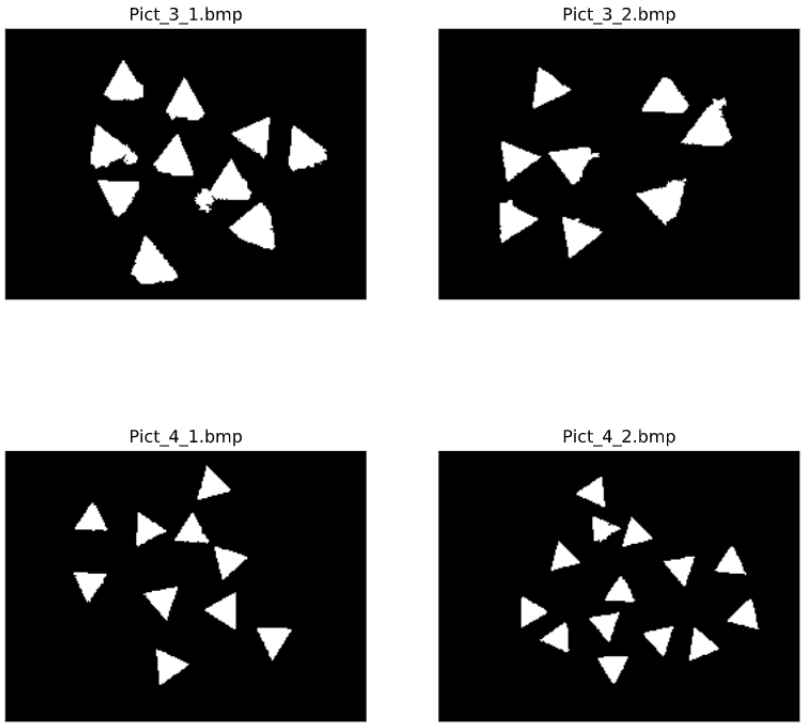

Рис. 5: Сегментационные маски для тестовой выборки

metric	exact match	ordinal match	set match
Pict_3_1.bmp	0.800	0.900	0.900
Pict_3_2.bmp	0.625	0.750	0.792
Pict_4_1.bmp	0.600	0.800	0.867
Pict_4_2.bmp	0.500	0.690	0.786
train set	**0.631**	**0.785**	**0.836**

Таблица 1. Метрики качества для предсказания маркировки фишек.

set match является наиболее объективной метрикой, а потому будем ориентироваться на неё. Возможны и дальнейшие улучшения показателей и обобщающей способности при помощи применения попиксельных преобразований.

Выводы

Таким образом предложенный алгоритм хорошо справляется с предложенной задачей и оставляет простор для улучшений. Задача сегментации изображения на тестовой выборке имеет максимальную точность, а задача классификации показывает приемлемый результат на предложенной структуре классов (**set match accuracy** = 0.836). Выполнение программы занимает относительно много времени (1 минута на изображение), однако это было сделано намеренно для повышения качества в задаче сегментации.

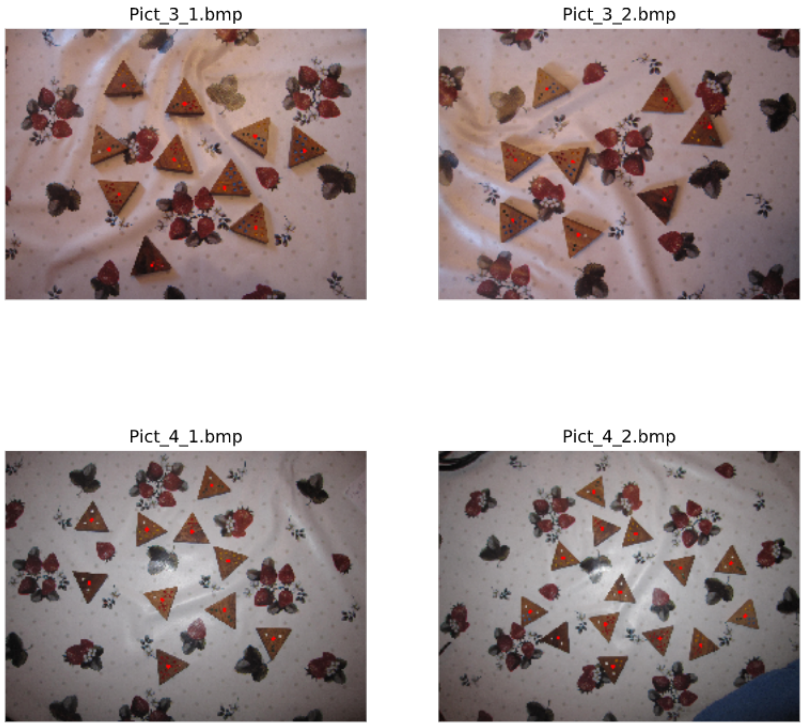

Рис. 6: Координаты центра для тестовой выборки