Нахождение области с текстом в изображении

Белорусский государственный университет информатики и радиоэлектроники
г.Минск, Республика Беларусь

Дубров И.В.
Телеш И.А. – канд. геогр. наук, доцент

Перед началом работы с распознаванием текста нужно максимально избавиться от всех лишних элементов на изображении и получить максимально сжатую область текста, с которой впоследствии производить все вычисления. Это сократит время работы программы и повысит качество распознавания.

Поэтому создание программы, которая может автоматически найти прямоугольник с текстом на изображении, является важным этапом при решении задачи распознавания текста.

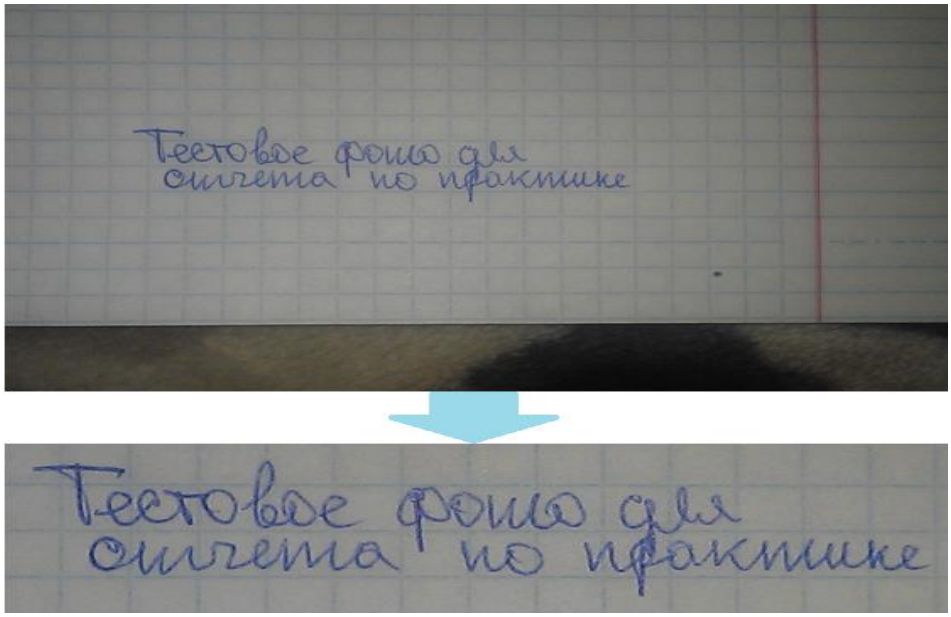

Рисунок 1 – Пример выделения текста из изображения

Для выделения текста из изображения были использованы следующие алгоритмы и пакеты:

- детектор границ «Canny» для обнаружения границ изображениями;
- rank-фильтр для удаления границ в 1 пиксель;
- F1-мера для решения задачи точности/полноты;
- реализация с помощью OpenCV, numpy, PIL [3].

Во-первых, применен детектор границ «Canny» к изображению. Это создает белые пиксели везде, где есть границы исходного изображения. Это удаляет большую часть фонового шума с изображения и превращает текстовые области в яркие скопления. Также превращает границы в длинные, четкие линии.

Источниками ребер в изображении являются границы и текст. Чтобы обнулить текст, необходимо устранить границы.

Один действительно эффективный способ сделать это с помощью rank-фильтра. Это существенно замещает пиксель чем-то вроде медианы пикселей слева и справа. Текстовые области имеют много белых пикселей, но границы состоят только из тонкой однопиксельной белой линии. Области вокруг границ будут в основном черными, поэтому rank-фильтр устранит их. Границы исчезают, но текст все еще остается. Несмотря на то, что это эффективно, он оставляет фрагменты текста за пределами границ. Это может быть хорошо для некоторых приложений, но я хотел бы устранить и их, потому что для поставленной задачи они совершенно неинтересны и могут только помешать более поздним операциям. Поэтому после применения rank-фильтра я нашел контуры в изображении. Это наборы белых пикселей, которые соединены друг с другом. Контуры границ легко выделить: это те области, чей ограничивающий прямоугольник покрывает большую часть изображения.

С полигонами для границ легко удалить все, что находится за пределами них. После этого остается изображение с текстом и, возможно, с некоторыми другими битами из-за пятен или следов на исходной странице.

На данный момент мы ищем зону (x1, y1, x2, y2), которая:

- максимизирует количество белых пикселей внутри ее;
- является как можно меньше.

Эти две цели находятся в противоречии друг с другом. Если мы возьмем все изображение, то мы охватим все белые пиксели, но мы полностью упустим цель 2: зона будет излишне большой.

Это проблема precision / recall:

- recall – доля белых пикселей внутри обрезаемого *прямоугольника;*
- *precision – это доля изображения вне прямоугольника.*

*Достаточно стандартным способом решения п*роблемы precision / recall является оптимизация F1-меры, гармонического среднего precision и recall. Это то, что нужно реализовать.

Набор всех возможных зон достаточно велик: W^2H^2, W и H – это ширина и высота изображения, но большинство зон не имеет большого смысла. Мы можем упростить задачу, найдя отдельные фрагменты текста. Чтобы сделать это, можно применить бинарное расширение к обведенному краю изображения. Это производит слияние белых пикселей друг в друга. Это повторяется несколько раз, пока не будет только нескольких связанных компонентов, которые представляют наибольший интерес [2].

Включая некоторые из этих компонентов и отвергая другие, можно сформировать хорошие потенциальные зоны. Теперь всплывает проблема суммирования подмножеств: какое подмножество компонентов производит зону, которая максимизирует F1-меру? Есть 2^N возможных комбинаций подмножеств, для исследования. Необходимо упорядочить компоненты по количеству белых пикселей, которые они содержат в исходном изображении, и продолжать добавлять компоненты, пока увеличивается F1-мера. Когда увеличение останавливается – конец [1].

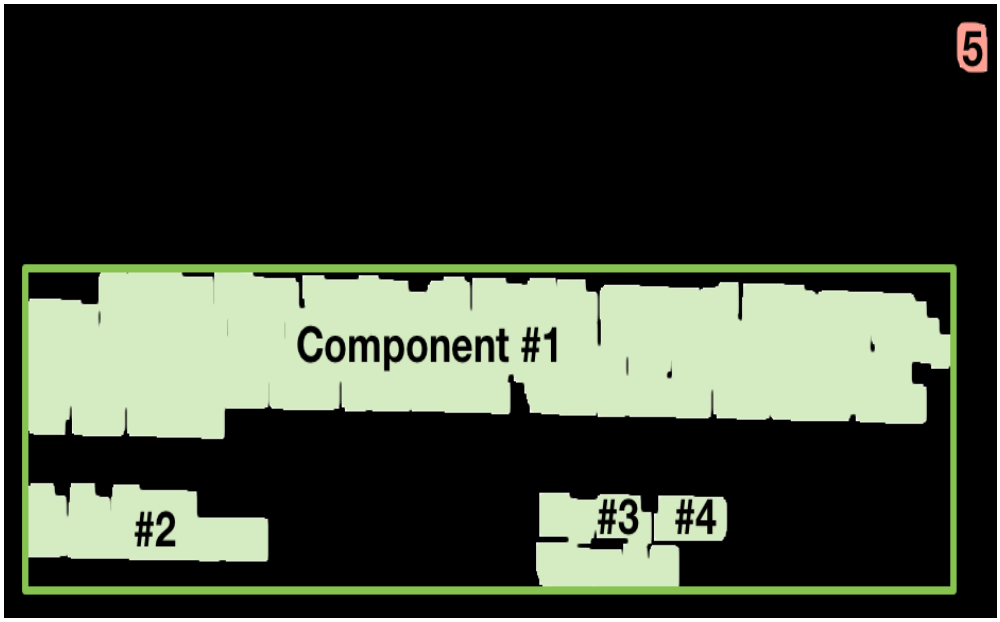

Рисунок 2 – Пример выделения зон с наибольшим содержанием текста

Компоненты упорядочены так, как описано выше: компонента с номером 1 содержит самое большое количество белых пикселей в исходном изображении. Первые четыре компоненты принимаются, а пятая отвергается, потому что она приводит к уменьшению F1-меры.

Описанная выше процедура хорошо работает во множестве случаев. Она выделяет правильные зоны, с наибольшим содержанием текста, примерно на 98% изображений, и все ошибки относительно незначительны.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Люггер, Д.Ф. Искусственный интеллект. Стратегии и методы решения сложных проблем / Д.Ф. Люггер. – М: Вильямс, 2003г. – 864 с.

2. Шапиро, Л. Компьютерное зрение / Л. Шапиро, Дж. Стокман. – М.: Бином. Лаборатория знаний, 2006. – 752 с.

3. OpenCV documentation [Электронный ресурс] – Минск, 2020. – Режим доступа: https://docs.opencv.org/3.0.0/. - Дата доступа: 19.04.2020.