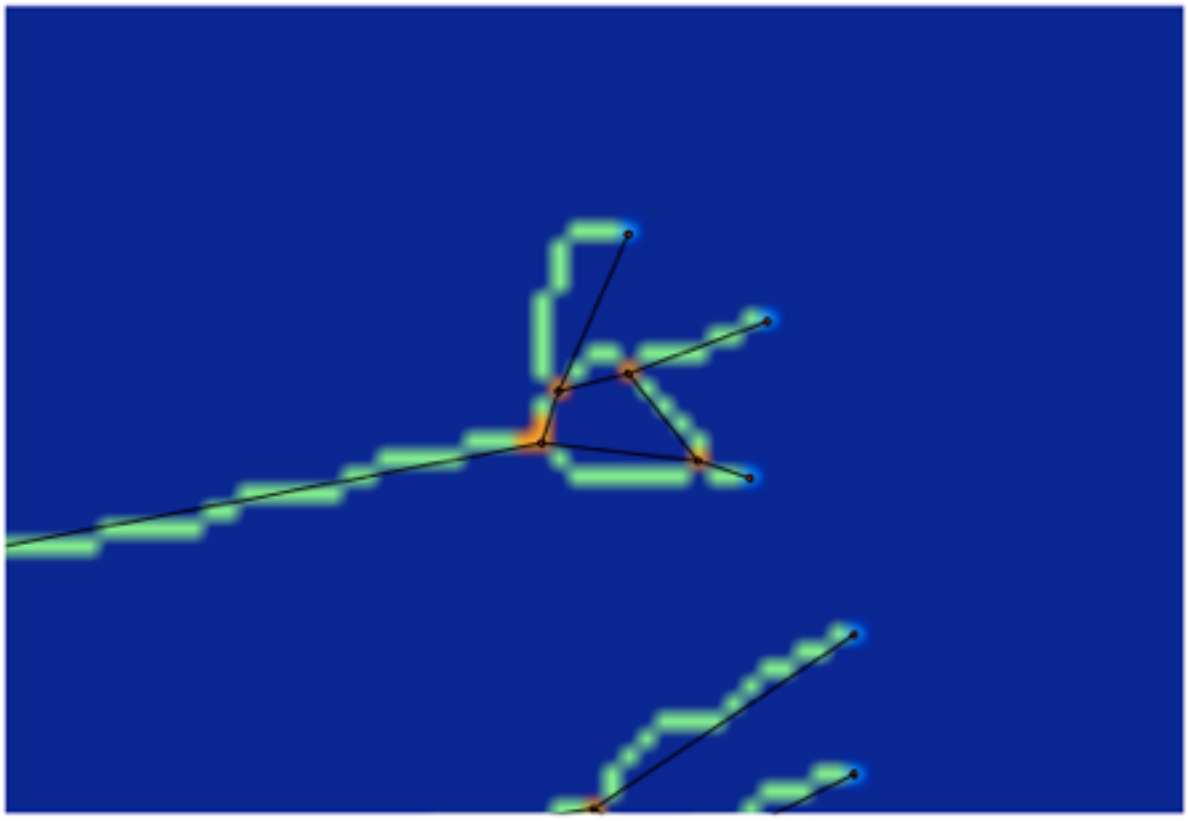
Анастасия Анциферова
520 группа
13 May 2017

Лабораторная работа №2 по курсу “Обработка и распознавание изображений” (Intermediate)



Постановка задачи

Необходимо разработать и реализовать программу для работы с изображениями геоглифов пустыни Наска. В процессе выполнения задания потребуется использовать методы распознавания формы объектов. В задании рекомендуется использовать построение скелета объекта, его обработка и извлечение признаков на основе полученных данных. Предлагается использовать один из алгоритмов построения скелета — Розенфельда, Зонг-Суня или использовать другие методы. Программа должна по данному изображению однозначно определять, какой из рисунков пустыни Наска на нем изображен.

Описание данных

На вход программе в уровне Intermediate подаются изображения геоглифов на белом фоне. Изображения содержат как оригиналы (оригинальная форма геоглифов), так и различные искажения - геометрические преобразования, имитирующие наблюдение геоглифа под разным углом. Необходимо провести бинаризацию данных изображений, построить скелет и найти уникальные признаки, с помощью которых удастся классифицировать изображения. Программа выдает файл, содержащий имя входного файла изображения и номер класса изображения (от 1 до 7). Входные изображения содержали в названии соответствующий метки классов.

Описание метода решения

Мною был реализован следующий алгоритм решения задачи:

1) Бинаризация изображения с помощью метода Оцу
2) Построение скелета с помощью метода skeletonize библиотеки scikit-image на Python
3) Нахождение точек ветвления скелета и крайних точек
    1) Обход всех пикселей изображения окном 3x3, проверка 8 соседей пикселя (x,y) и определение наличия ветвления по их числу
    2) Добавление координат точки в граф, реализованный с помощью библиотеки networkx на Python
4) Нахождение ребер скелета
    1) Для всех точек ветвления скелета (вершин) выполнялся рекурсивный алгоритм
    2) В окрестности пикселя рассматривались 8 соседей точки и выбиралось направление обхода
    3) Алгоритм останавливался, когда находил другую точку ветвления

4) В процессе обхода вычислялась длина ребра скелета и сохранялась в параметрах графа

5) Фильтрация данных графа скелета
   1) Объединение рядом лежащих вершин графа в одну, к которой присоединялись все ребра от объединенных вершин
   2) Удаление петель
   3) Удаление ребер, которые появлялись из-за утолщения геоглифа - небольшие “отростки” скелета, которе можно было охарактеризовать следующим образом: вершина, у которой 3 ребра, одна ребро короткое и имеет на втором конце терминальную вершину (отросток) и два других длинные и не имеют терминальные вершины на других концах
   4) Удаление изолированных вершин, которые могли появиться из-за предыдущих шагов или бинаризации
6) Выделение признаков графа скелета
   1) количество связных компонент
   2) количество ребер
   3) количество терминальные вершин
   4) максимальная степень вершины
   5) количество вершин и ребер, степени и длины больше определенного числа
7) Создание меток картинок по и названию (Силуэт_5_… -> метка 5)
8) Обучение классификатора (k ближайших соседей, использовался 1 сосед) на 4 изображениях, которые не содержали искажений (не имели в названии приставок №: “Силуэт_1_№”)
9) Проверка точности классификатора на оставшихся 3 искаженных изображениях для каждого класса

Описание программой реализации

При реализации алгоритма использовались библиотеки skimage - для начальной обработки изображений, networkx для построения, хранения и обработки графа скетела, scikit-learn для обучения и тестирования классификатора. Объем реализации составил 379 строк.
При создании программы была произведена попытка реализовать алгоритм Розенфельда, однако скорость работы реализации на языке Python была низкой, поэтому было отдано предпочтение методу skeletonize3d, основанному на алгоритме Ли 1994 года.

Запуск программы:

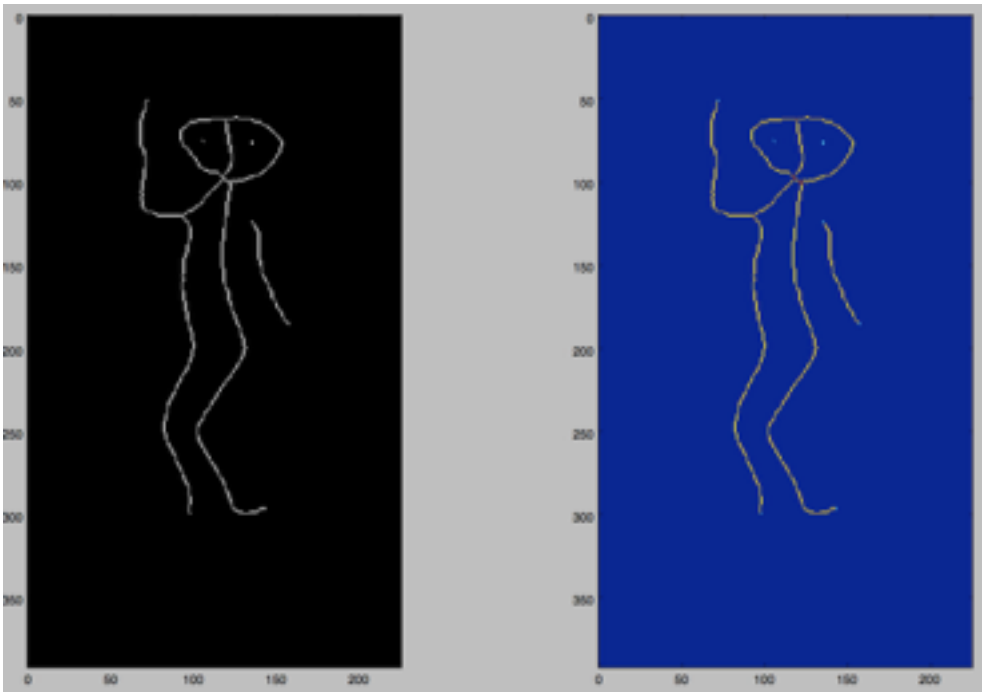
В main программы ввести папку, содержащую изображения, среди которых есть исходные изображения

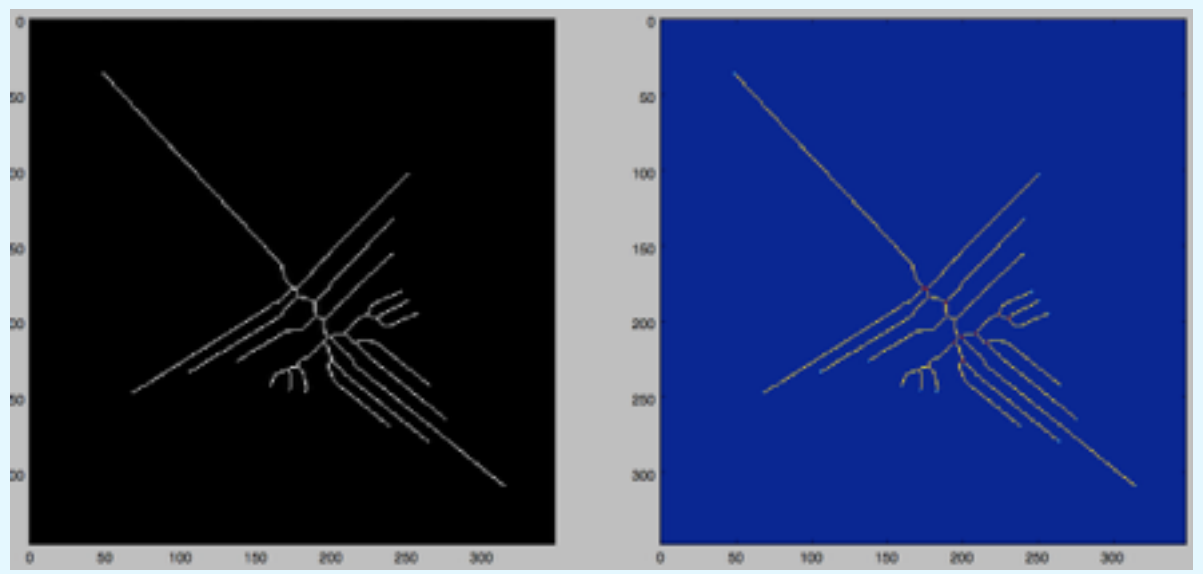
path = 'Geogliph_1'
Ввести имя папки, куда сохраняется облачный вывод - графы и скелеты геоглифов
out_path = ‘Result'
Написать номера изображений, на которых будет обучаться классификатор. Это могут быть изображения, данные вместе с заданием. В их названиях содержится метка класса. На этих и остальных изображениях будет проводится тестирование классификатора. По умолчанию для обучения заданы номерах исходных изображение географов (без искажений).
list_train_images = [0,4,8,12,16,20,24]
Программа выдаст точность работы классификатора на всей выборке изображений из заданной папки.

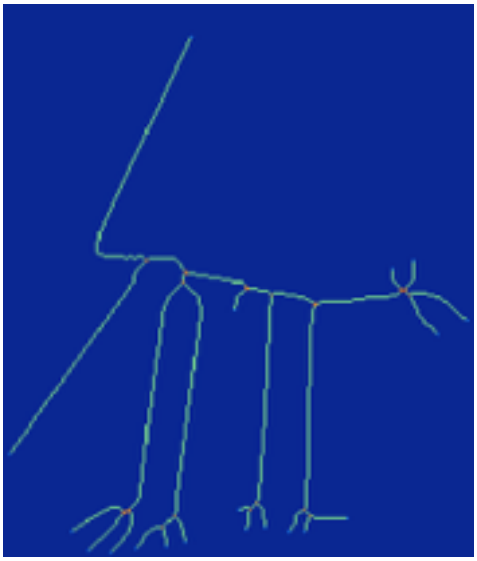
Эксперименты

1. Построение скелета и поиск вершин

На левом изображении показан скелет, построенный с помощью skeletonize и мой алгоритм поиска точек ветвления скелета. Цвет точки показывает количество её соседей в блоке 3x3, на правом изображении подсвечены точки ветвления скелета.





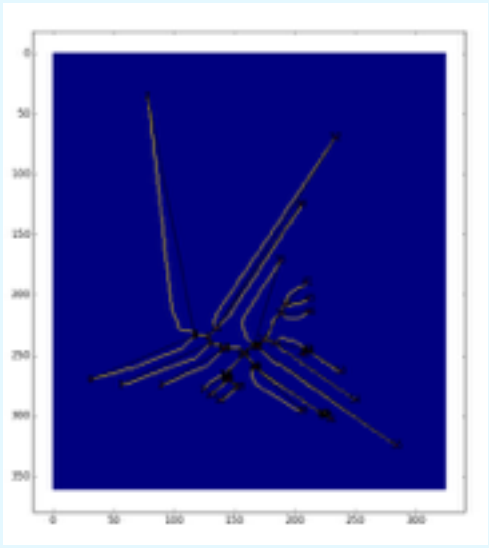


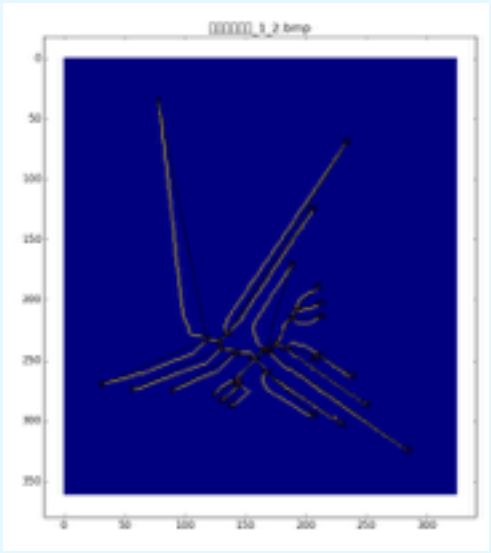
2. Построение графа скелета, фильтрация графа

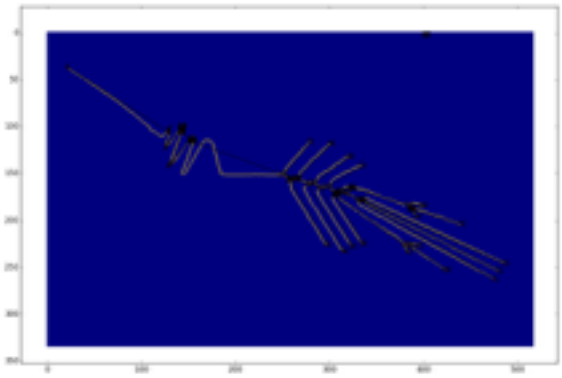
На изображениях показан граф скелета поверх самого скелета с сохранением координат вершин. На левой изображении - граф после построения его путем обхода всех ребер из точек ветвления, на правом - после обработки. В графе удалены вершины, находящиеся близко друг от груда, грани - петли, изолированные вершины, терминальные вершины, находящиеся близко к другим терминальным вершинам и имеющие короткое ребро с соседней вершиной, если она не терминальная и не имеет с других коротких ребер. Также в графе удалены вершины и ребра, которые имели 2 ветвления и по сути повторяли изгиб скелета, который мог образоваться после деформации геоглифа.

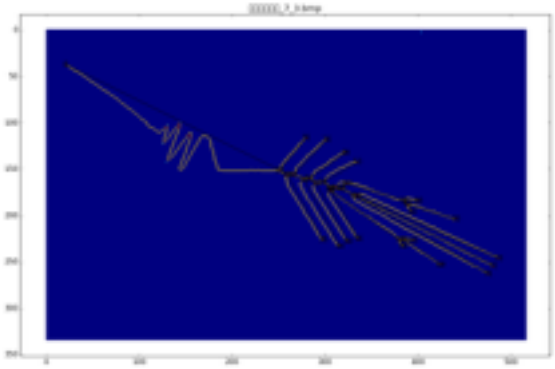
Оригинал графа скелета

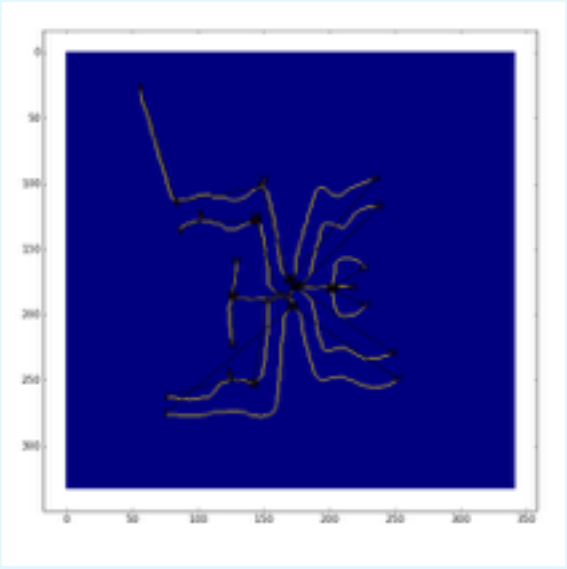
После обработки

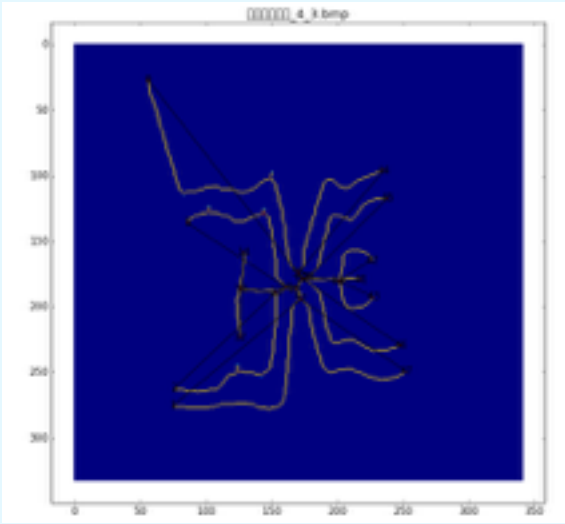


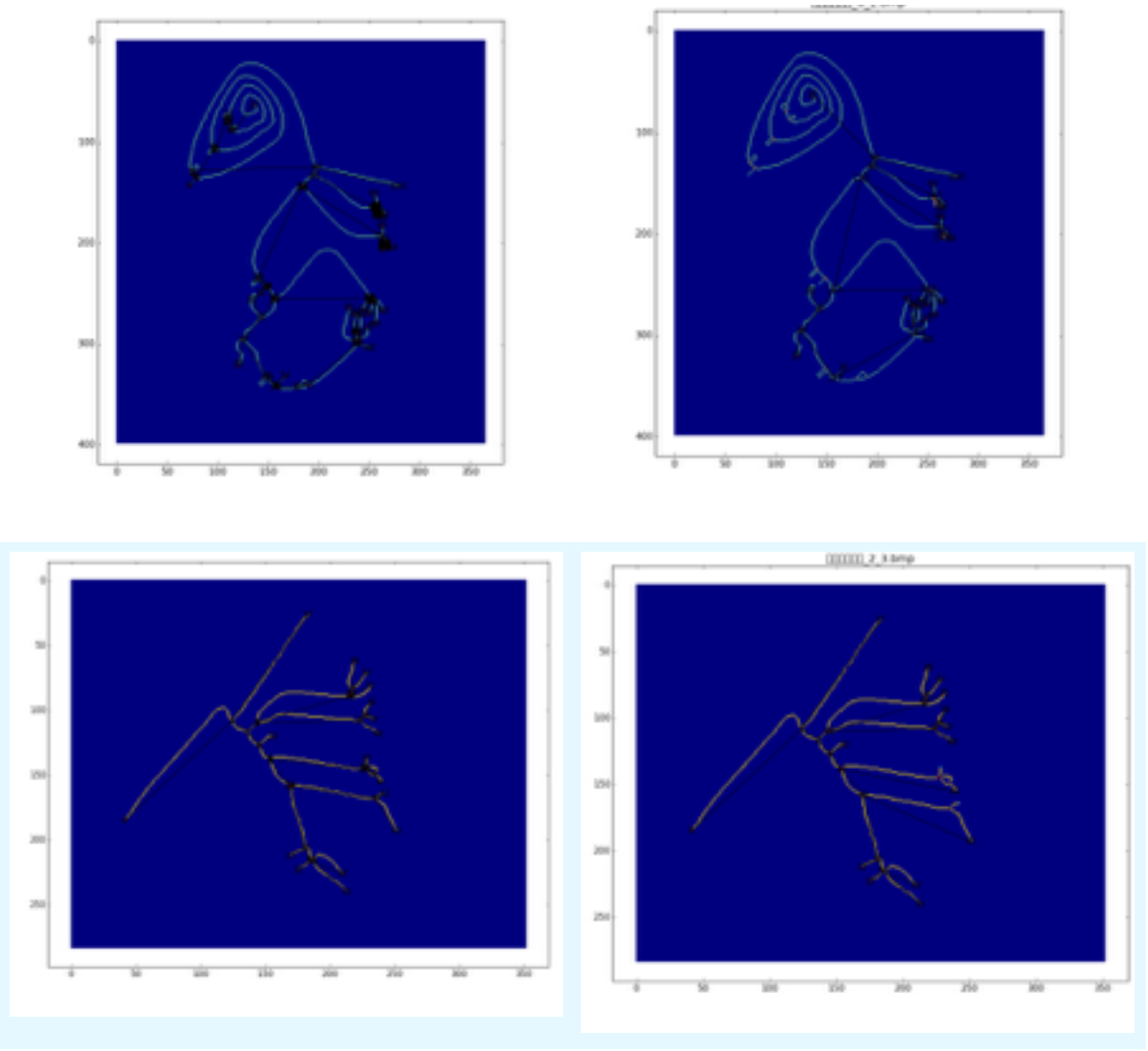








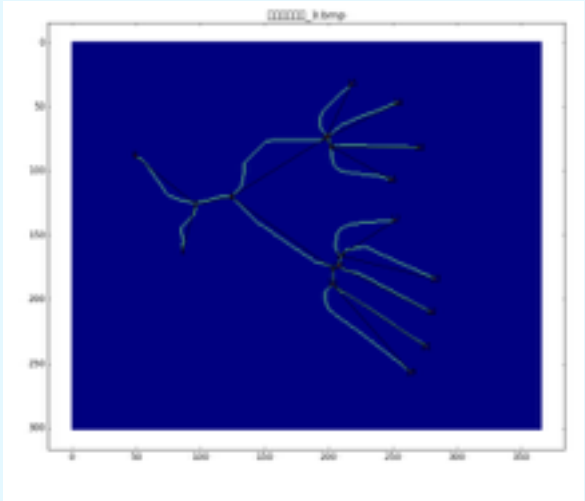




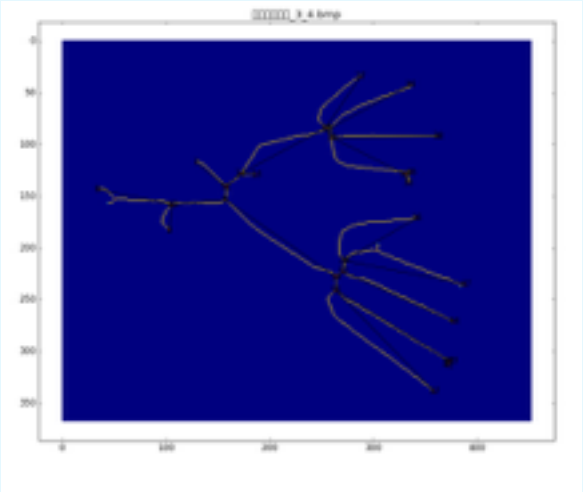
Выводы

В процессе разработки было протестировано много модификаций алгоритма, и лучшая из них работает с точностью 96% (после обучения на 4 оригинальных изображениях среди всех остальных алгоритм ошибается только на 1 изображении). Данное изображение из-за значительного искажения меняет форму скелета следующим образом:

Оригинал



Изображение 3_4



Как видно на изображениях скелета, у искаженного рисунка появляется несколько лишних терминальных вершин, при этом длина ребер между ними и соседними вершинами достаточно длинная и не удаляется при отбрасывании коротких ребер.

В процессе выполнения задания я больше узнала о скелетизации объектов, встретилась с трудностями этой задачи. Несмотря на то, что существуют готовые библиотеки с методами построения скелета, эти методы не идеальны и полученный скелет требует дальнейшей обработки. Я так же попробовала реализовать на языке Python алгоритм Розенфельда из оригинальной статьи, однако, моя реализация работала достаточно медленно, поэтому я решила использовать метод из библиотека scikit-image. В нем реализуется алгоритм Ли 1994 года (T.-C. Lee, R.L. Kashyap and C.-N. Chu, Building skeleton models via 3-D medial surface/axis thinning algorithms. Computer Vision, Graphics, and Image Processing, 56(6):462-478, 1994.) Также мне удалось улучить свой навык работы с библиотекой networkx для построения графа полученного скелета. Полученная структура позволила мне удобно хранить информацию о структуре изображения, и с его помощью можно получить и использовать много различных полезных признаков изображения.