Определение береговой линии водоёма по спутниковым снимкам

Гаев Владислав Горбанева Олеся Игнатов Платон
Исангулов Александр Пронькин Алексей Щеглов Артём
Эркинбекова Айжамал Янковский Фёдор Авзалов Дмитрий

1 Введение

Береговая линия является кривой пересечения суши и некоторого водного объекта. Эта граница описывается контуром, представляющим из себя конечное множество точек. Ручная разметка спутникового снимка требует много времени, поэтому, чтобы решить данную проблему, необходимо разработать алгоритм автоматической сегментации водоёмов, основанный на методах компьютерного зрения.

2 Постановка задачи

Проблематика. Заказчик проекта размечает береговые линии на карте в приложении QGIS — свободной кроссплатформенной геоинформационной системе, позволяющей создавать и редактировать данные, производить карты, а также выполнять аналитические операции. На ручную качественную разметку одной карты может потребоваться несколько часов, а в некоторых случаях и несколько дней.

Образ результата. Требуется разработать расширение для приложения QGIS, с помощью которого пользователь сможет получить на выходе векторный слой маски, отражающей сегмент водного объекта на заданной области.

3 Методы решения задачи

3.1 Классические алгоритмы компьютерного зрения

Основная проблема классических методов заключается в сложности их обобщения для произвольного снимка. Главным недостатком каждого из нижеприведённых алгоритмов является сильная зависимость от свойств входного изображения.

3.1.1 Multi-layer adaptive threshold + Watershed

Данный алгоритм основывается на гипотезе, что разница между соседними пикселями водных объектов мала.
Результат обработки таким методом зависит от двух параметров:

- Наличие плотного шума на **неводных** объекте.
- Однородность пикселей на водном объекте.

Иными словами, допустим $P(x, y)$ — интенсивность пикселя в точке (x, y), и K — некоторое пороговое значение, тогда для первого случая:

$$S_{x,y} = \frac{\sum_{i=x-1}^{x+1} \sum_{j=y-1}^{y+1} P(x, y) - P(i, j)}{8}, \quad S_{x,y} \gg K$$

Для второго случая, соответственно:

$$S_{x,y} \leq K$$

Сам алгоритм представляет из себя последовательное обрабатывание исходного изображения методами компьютерного зрения, предоставленными библиотекой OpenCV:

1. Gaussian blur ($\sigma_y = 2, \sigma_x = 2, ksize = (11, 11)$)
2. Adaptive threshold ($ksize = 11, C = 2$) + OTSU threshold
3. Erode ($iterations = 1$) $\circ$ Dilation ($iterations = 3$)
4. Morphological closing ($iterations = 10$) $\circ$ global threshold ($thresh = 128$)
5. Distance transform (mode = L2) $\circ$ normalization $\in$ [0, 1]
6. Вычисление оптимального порога k = argmax ($\nabla dist.hist()$) $\circ$ distance threshold (thresh = k)
7. Result = Distance & closing

Пример последовательной обработки изображения алгортмом показан ниже:

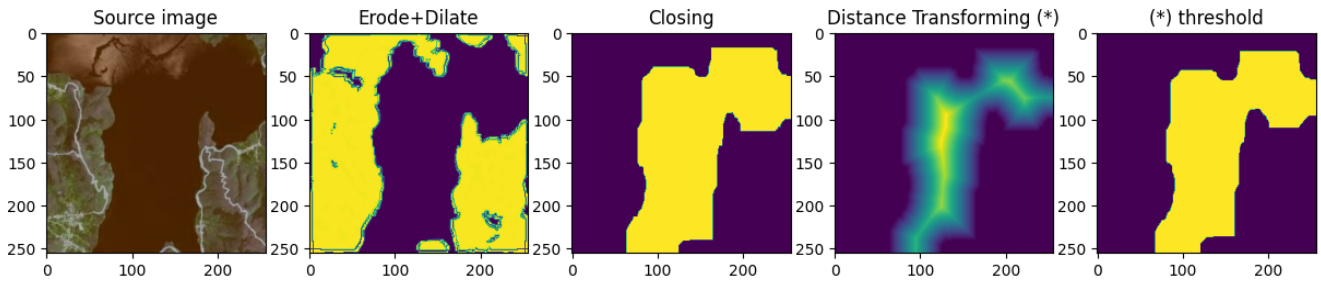

Рис. 1: Watershed алгоритм

На изображении видно, что в результате второй операции лево-верхняя часть водоёма была обработана как фон: это произошло из-за наличия такой шумной области, как облако. Помимо этого, причиной некачественной обработки снимка могут послужить такие артефакты, как рябь и пересечение двух или более лоскутов. (см. рис. 2)

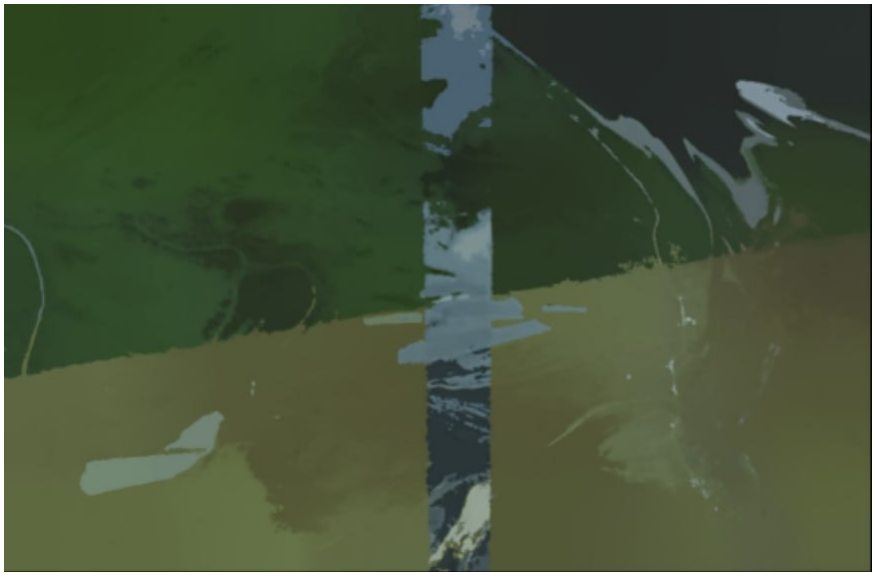

Рис. 2: Пример пересечения лоскутов на снимке

3.1.2 FloodFill

Помимо "многослойной бинаризации"существует ряд других алгоритмов сегментации, одним из которых является FloodFill. Данный метод более гибкий, чем Watershed, но по существу мало чем отличается от него. Он также сегментирует объекты на изображении по принципу однородности, однако в отличие от предыдущего алгоритма анализирует значения пикселей не в чёрно-белом формате, а в HSV палитре.

- **Преимущества**
  1. Позволяет сегментировать произвольный объект, указав точку на изображении
  2. Намного лучше работает на более простых изображениях, чем Watershed.
- **Недостатки**
  1. Алгоритм достаточно плохо экстраполируется на сложных снимках.
  2. Требует интервал цвета в HSV палитре.
  3. Проблематично модифицировать внутреннюю реализацию OpenCV

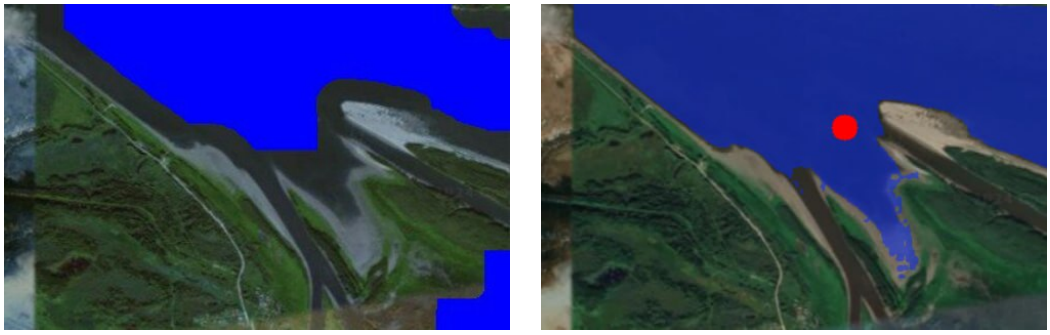

Рис. 3: Сравнение алгоритмов сегментации: Watershed (слева) и FloodFill (справа)

Заключение. На вопрос, что лучше использовать: Watershed или FloodFill алгоритм, сначала может показаться, что правильным будет ответить второй метод, так как в данной задаче точность обработки имеет больший вес, чем возможность обобщаться на произвольное изображение с более низкими значениями метрики. С другой стороны, необходимость в уточнении интервала цвета в HSV палитре будет весомым отрицательным аспектом для пользователя, так как диапазон значений заранее предопределить невозможно: не всякий водоём имеет синий оттенок, бывают также серые и зеленые тона (с последними возникают еще бОльшие проблемы из-за наличия лесов на фоне).

Недостатки обоих алгоритмов делают бесполезным их использование на практике, поэтому необходимо разработать или использовать такой метод/модель машинного обучения, который(-ая) обладал(-а) бы высокой точностью и требовал(-а) от пользователя на вход минимальное количество параметров.

3.2 Нейросетевые модели

Проблема отсутствия стабильности при качественной обработке снимков классическими методами компьютерного зрения может быть решена при помощи алгоритмов глубокого обучения — нейросетей. При таком подходе обобщающаяся способность модели в основном ограничена лишь её видом и обучающей выборкой. В рамках данной задачи последний пункт имеет отдельное внимание, так как заказчиком не были предоставлены какие-либо данные, удовлетворяющие входным параметрам моделей.

Датасет. В качестве обучающей выборки были использованы данные с сайта Kaggle, которые содержали в себе две папки — снимки и маски. Предположительно, этот датасет был составлен не вручную, а при помощи классических методов КЗ, так как большинство масок содержало шум, который, в свою очередь, может негативно повлиять на качество обучения модели. Эта проблема была частично решена при помощи морфологических операций — эрозии и делеции.

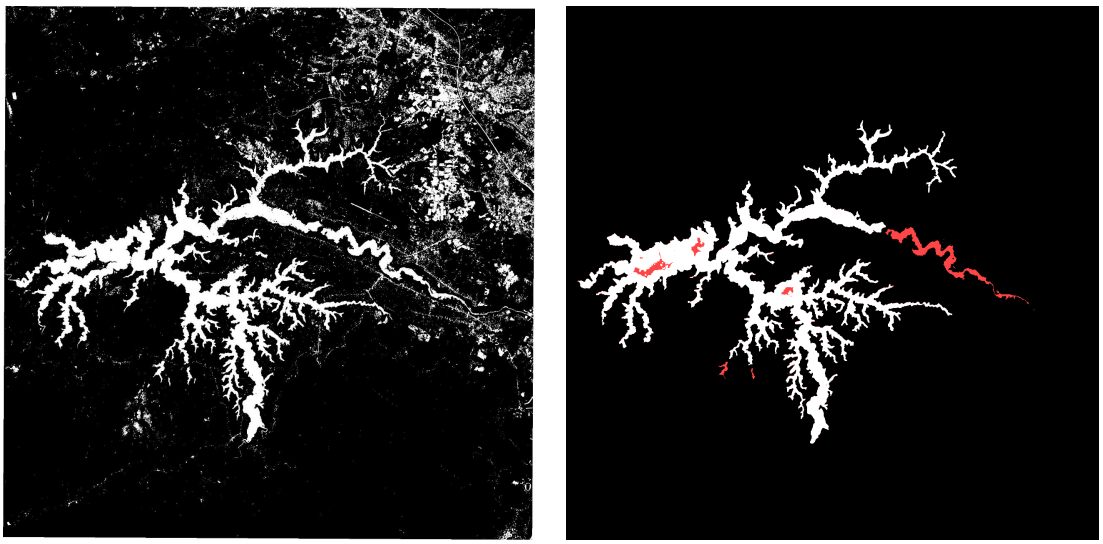

Рис. 4: Предобработка маски при помощи эрозии и делеции. На втором изображении красным обозначены утраченные сегменты.

Решена частично потому, что эта предобработка имела побочный эффект в виде удаления некоторой области исходных водоёмов, однако это должно несильно повлиять на качество модели, так как далеко не на каждой маске были удалены нужные сегменты, за счёт чего модель может экстраполироваться.

Помимо предобработки, при помощи сервиса Roboflow была также применена аугментация к датасету, состоящая из двух операций: flip и hue. В итоге размер данных был увеличен примерно в 1.75 раз (4922 снимков).

3.2.1 YOLOv8

Была выбрана достаточно небольшая версия модели **yolov8m-seg**, принимающая на вход 27M параметров. Обучение проводилось на двух видеокартах Tesla T4. В качестве гиперпараметров были использованы $batch = 32$ и $imgsz = 640$.

В результате 50-ти эпох значение seg-loss дошло до отметки 0.748, а значение метрики $mAP50 - 95$ до 0.87. В целом на обучение ушло 108 минут.

3.2.2 SAM-tiny

Также была дообучена модель SAM-tiny. Дообучение было на пятнадцати эпохах, batch=8. Значения метрик для данной модели указаны ниже.

4 Метрики

Для сравнительного анализа будем использовать метрику IoU (Intersection over Union), которая определяется как:

$$IoU = \frac{TP}{TP + FN + FP}$$

Тогда искомые таблицы значений будут выглядеть так:

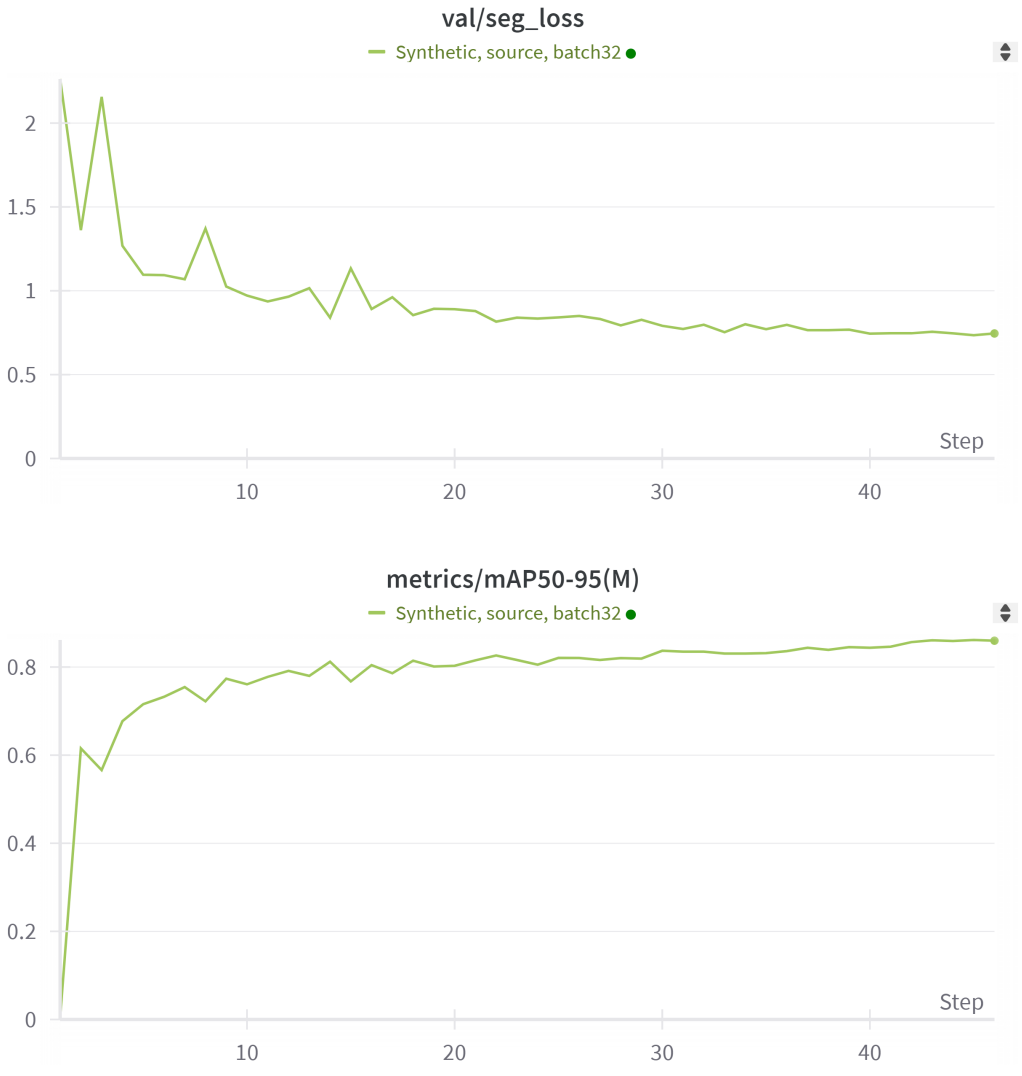

Рис. 5: Обучение модели YOLOv8

YOLOv8 (без обработки датасета)	YOLOv8 (с обработкой)	YOLOv8 (с обработкой и синтетикой)
0.546	0.813	**0.917**

SAM-tiny (без обучения с обработкой)	SAM-tiny (с обучением и обработкой)
0.631	**0.845**

Watershed	FloodFill
0.540	**0.553**

Как видно из таблицы, классические методы очень сильно проигрывают нейронным сетям, что в принципе было очевидным. Интереснее обстоят дела между лучшими YOLOv8 и SAM моделями: у первой достаточно большой отрыв от второй — 0.072. Однако стоит учитывать тот факт, что YOLO модель, помимо исходного датасета, была обучена ещё на синтетических данных (на данный момент алгоритм генерации синтетических данных находится в разработке; подробнее можно узнать в следующем разделе), поэтому разумнее будет сравнивать значения 0.813 и 0.845. Несмотря на всё это, синтетические данные являются лишь дополнением к основным, а не их заменой, поэтому модель YOLOv8 фактически можно назвать лучшей в этой таблице.

5 В разработке

5.1 Синтетические данные

Мотивом к созданию генератора синтетических данных послужило низкое разрешение изображений в датасете и потеря нужных сегментов водоёмов. Сейчас алгоритм по праву можно назвать классическим, так как он не содержит в себе методов глубокого обучения, однако в будущем планируется создать генератор на основе диффузионной модели.

Процесс создания состоит из двух этапов: генерации водоёма на основе маски и, соответственно, генерации леса.

Генерация водоёма. В основе идеи лежит функция шума Перлина.

Шум Перлина — это градиентный шум, состоящий из набора псевдослучайных единичных векторов (направлений градиента), расположенных в определенных точках пространства и интерполированных функцией сглаживания между этими точками. Для генерации шума Перлина в одномерном пространстве необходимо для каждой точки этого пространства вычислить значение шумовой функции, используя направление градиента (или наклон) в указанной точке. [?]

В качестве функции сглаживания берётся линейная интерполяция:

$$f(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0), x \in [x_0, x_1]$$

Будем использовать нормально распределенное значение размера ячейки ($\mu = 400, \sigma = 20$), чтобы не симулировать большое количество артефактов на изображении.

Рис. 6: Пример сгенерированной воды

Генерация леса. Здесь идея несколько похожа на свёртку картинок: будем генерировать Гауссовский шум на изображениях разных размеров (8x8, 32x32 и 512x512), а затем увеличивать/уменьшать их при помощи кубической интерполяции. Результатом будет морфологическая операция Tophat на объединении этих изображений.

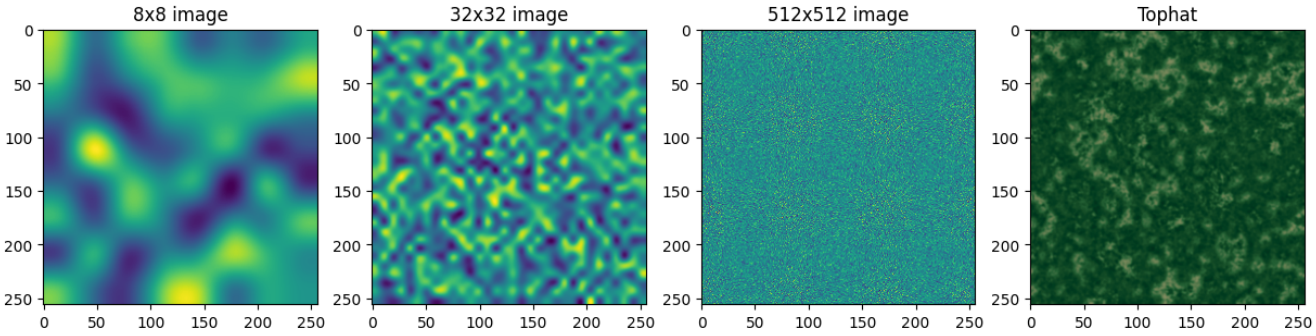

Рис. 7: Пример сгенерированного леса

Результатом всего алгоритма является объединение двух слоёв на основе исходной маски.

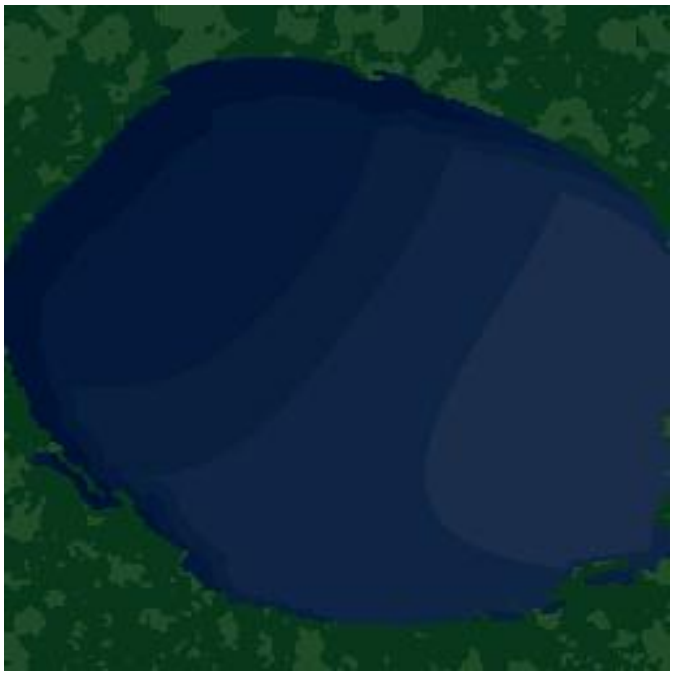

Рис. 8: Результат генерации

5.2 Создание графического интерфейса

Приложение QGIS даёт разнообразный функционал для разработки визуальной части приложения. Из всех предложенных вариантов был выбрана отрисовка плагина в отдельном окне.

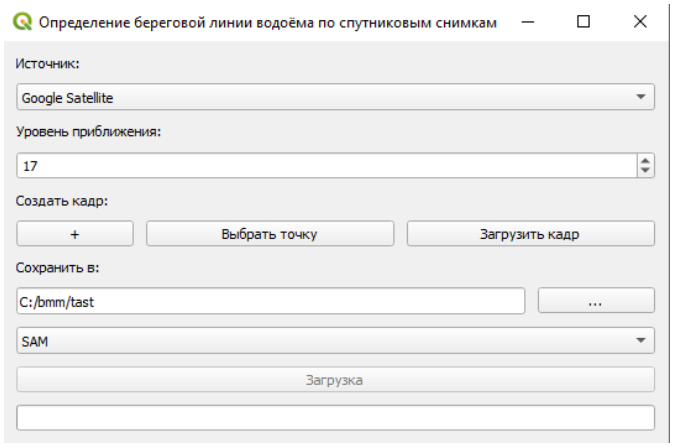

Рис. 9: Окно плагина

Источник. Оборудование на спутниках не идеально, и это может повлиять на качество некоторых снимков. От того, какой источник данных используется, зависит точность определения береговой линии.

Уровень приближения. Чтобы выбрать масштаб для отрисовки, нужно учитывать, что чем больше масштаб, тем более детализированным и качественным получается снимок.

Создать кадр. Инструмент, который позволяет создавать кадры и выбирать начальную точку для моделей и алгоритмов, требующих этого.
Путь к папки нужен для сохранения получившихся снимков
Список моделей и алгоритмов позволяет выбрать наиболее подходящий вариант для конкретной местности и устройства, где будет запускаться плагин.
Загрузка — запуск плагина.