**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**

**Кафедра информационных систем управления**

**РАЗРАБОТКА СИСТЕМ АНАЛИЗА ДАННЫХ ДИСТАНЦИОННОГО ЗОНДИРОВАНИЯ НА ОСНОВЕ ТЕХНОЛОГИИ OPENLAYERS.**

Курсовая работа

Кулешевича Тимофея Витальевича

студента 3 курса,

специальность «прикладная информатика»

Научный руководитель:

доцент кафедры

О.Л. Коновалов

Минск, 2023

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc135579380)

[1. Технология OpenLayers 4](#_Toc135579381)

[1.1. Основные возможности OpenLayers 4](#_Toc135579382)

[1.2. Установка OpenLayers 6](#_Toc135579383)

[2. Система анализа данных дистанционного зондирования 7](#_Toc135579384)

[2.1 Постановка задачи дистанционного зондирования 7](#_Toc135579385)

[2.2. Архитектура проекта 7](#_Toc135579386)

[2.2.1 Карты 8](#_Toc135579387)

[2.2.2 Координаты 9](#_Toc135579388)

[2.2.3 Сохранение карт 10](#_Toc135579389)

[2.2.4 Выделение области поиска теплиц 11](#_Toc135579390)

[2.2.5 Разбиение участка на тайлы. 12](#_Toc135579391)

[2.2.6 Выделение контуров 13](#_Toc135579392)

[2.2.7 Установка точек на карте 14](#_Toc135579393)

[2.2.8 Ручной режим 15](#_Toc135579394)

[3. Разработка ПО на основе OpenLayers 17](#_Toc135579395)

[3.1. Index.html 17](#_Toc135579396)

[3.2. Main.js 18](#_Toc135579397)

[3.3 Styles.css 28](#_Toc135579398)

[ЗАКЛЮЧЕНИЕ 30](#_Toc135579399)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 31](#_Toc135579400)

# ВВЕДЕНИЕ

Дистанционное зондирование является одним из наиболее эффективных методов получения информации о земной поверхности и атмосфере. Современные технологии дистанционного зондирования позволяют получать большое количество данных, которые могут быть использованы для решения различных задач, связанных с изучением окружающей среды и ее изменениями. Однако, обработка и анализ такого объема данных является сложной задачей, требующей использования специализированных инструментов и технологий.

Современные методы дистанционного зондирования позволяют получать большое количество данных, которые могут быть использованы для решения различных задач, связанных с изучением окружающей среды и ее изменениями. Однако, для эффективного использования этих данных необходимы специализированные инструменты и технологии, которые обеспечат быстрый и удобный доступ к информации, ее анализ и визуализацию.

Целью данной работы является разработка системы анализа данных дистанционного зондирования. Основное предназначение этой системы заключается в автоматизации обработки снимков участка земной поверхности. Результаты работы программы могут быть использованы для обучения нейронных сетей или для анализа полученных данных с целью получения информации о географический объектах.

Данная работа может быть полезна для специалистов в области дистанционного зондирования, а также для научных и исследовательских организаций, занимающихся изучением окружающей среды и ее изменениями, а также в сельском хозяйстве, геологии и других областях применения.

# 1. Технология OpenLayers

В данной работе рассматривается разработка системы анализа данных дистанционного зондирования на основе технологии OpenLayers. Для работы с OpenLayers вам понадобятся базовые знания web-программирования, JavaScript, HTML и CSS.

OpenLayers является мощной библиотекой JavaScript для отображения карт и визуализации геоданных в веб-браузере. Библиотека позволяет работать с различными типами геоданных, включая дистанционные снимки, и обеспечивает широкий спектр возможностей для их анализа и визуализации. Это полностью бесплатный JavaScript с открытым исходным кодом, выпущенный под лицензией BSD (также известной как FreeBSD).

OpenLayers позволит обеспечить эффективное использование данных дистанционного зондирования в различных областях, таких как экология, сельское хозяйство, геология и многие другие.

## 1.1. Основные возможности OpenLayers

OpenLayersпредоставляет множество различных возможностей использования географической информации всех видов.

* Отображение различных типов карт:

OpenLayers позволяет отображать различные типы карт, включая фрагментные(tile) карты, карты с векторными данными и карты, основанные на различных источниках данных.

* Работа с геоданными:

OpenLayers предоставляет возможность работы с различными типами геоданных, включая геометрические объекты, географические координаты, атрибутивные данные и многие другие.

* Интерактивность:

OpenLayers позволяет создавать интерактивные карты и приложения, которые позволяют пользователям взаимодействовать с картой, изменять ее масштаб, перемещать и выбирать объекты.

* Анализ данных:

OpenLayers обеспечивает возможность анализа геоданных и создания моделей на их основе.

* Поддержка различных проекций:

OpenLayers поддерживает большое количество проекций, что позволяет работать с геоданными, полученными в различных системах координат.

* Взаимодействие с другими библиотеками и сервисами:

OpenLayers интегрируется с другими библиотеками и сервисами, такими как GDAL, GeoServer, PostGIS, Leaflet и другими.

* Создание пользовательских элементов управления:

OpenLayers позволяет создавать пользовательские элементы управления, такие как кнопки, поля ввода и другие элементы, которые могут быть добавлены на карту.

* Поддержка мобильных устройств:

OpenLayers поддерживает работу на мобильных устройствах, что позволяет создавать мобильные приложения для работы с геоданными.

* Расширяемость:

OpenLayers является расширяемой библиотекой, что позволяет создавать пользовательские модули и расширения для решения специфических задач.

* Открытый исходный код:

OpenLayers распространяется под лицензией BSD, что позволяет использовать и модифицировать код библиотеки в своих проектах без ограничений.

## 1.2. Установка OpenLayers

Чтобы создать проект с и использованием OpenLayers нужно выполнить несколько шагов.

1. Установите Node.js и npm на свой компьютер, если они еще не установлены.
2. Откройте терминал или командную строку и перейдите в папку, где вы хотите создать свой проект.
3. Введите команду npm init.

Эта команда попросит вас ввести информацию о вашем проекте, такую как название, версию и описание (можете нажимать Enter, чтобы оставить поля пустыми).

1. Установите библиотеку OpenLayers, выполнив команду npm install ol.
2. Установите vite (инструмент для сборки и запуска проекта, используемый в OpenLayers) командой npm install vite.
3. В папке проекта должен был создаться файл package.json.

Откройте его и добавьте строки в ячейку scripts, чтобы получилось:

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1",

"start": "vite",

"build": "vite build"

},

1. Добавьте index.html, main.js и styles.css файлы.

Содержимое вы можете взять из примеров в интернете: <https://openlayers.org/doc/quickstart.html> или <https://openlayers.org/en/latest/examples/>.

1. Запуск и сборка проекта.

Чтобы запустить проект нужно ввести команду npm start. Чтобы собрать проект нужно ввести команду npm build (собранный проект появиться в папке dist).

# 2. Система анализа данных дистанционного зондирования

## 2.1 Постановка задачи дистанционного зондирования

Рассмотрим решение прикладной задачи дистанционного зондирования с использованием OpenLayers.

Заказчик является крупной конторой, специализирующейся на починке теплиц. Аналитики этой конторы выяснили, что лучше всего работает целевая реклама, т.е. когда рекламная брошюра приходит прямо по адресу потенциального клиента. В связи с этим появилась необходимость поиска на участке карты всех теплиц. В результате заказчик должен получить список всех найденных теплиц, их адреса и их площади (для определения релевантности и объема работ).

## 2.2. Архитектура проекта

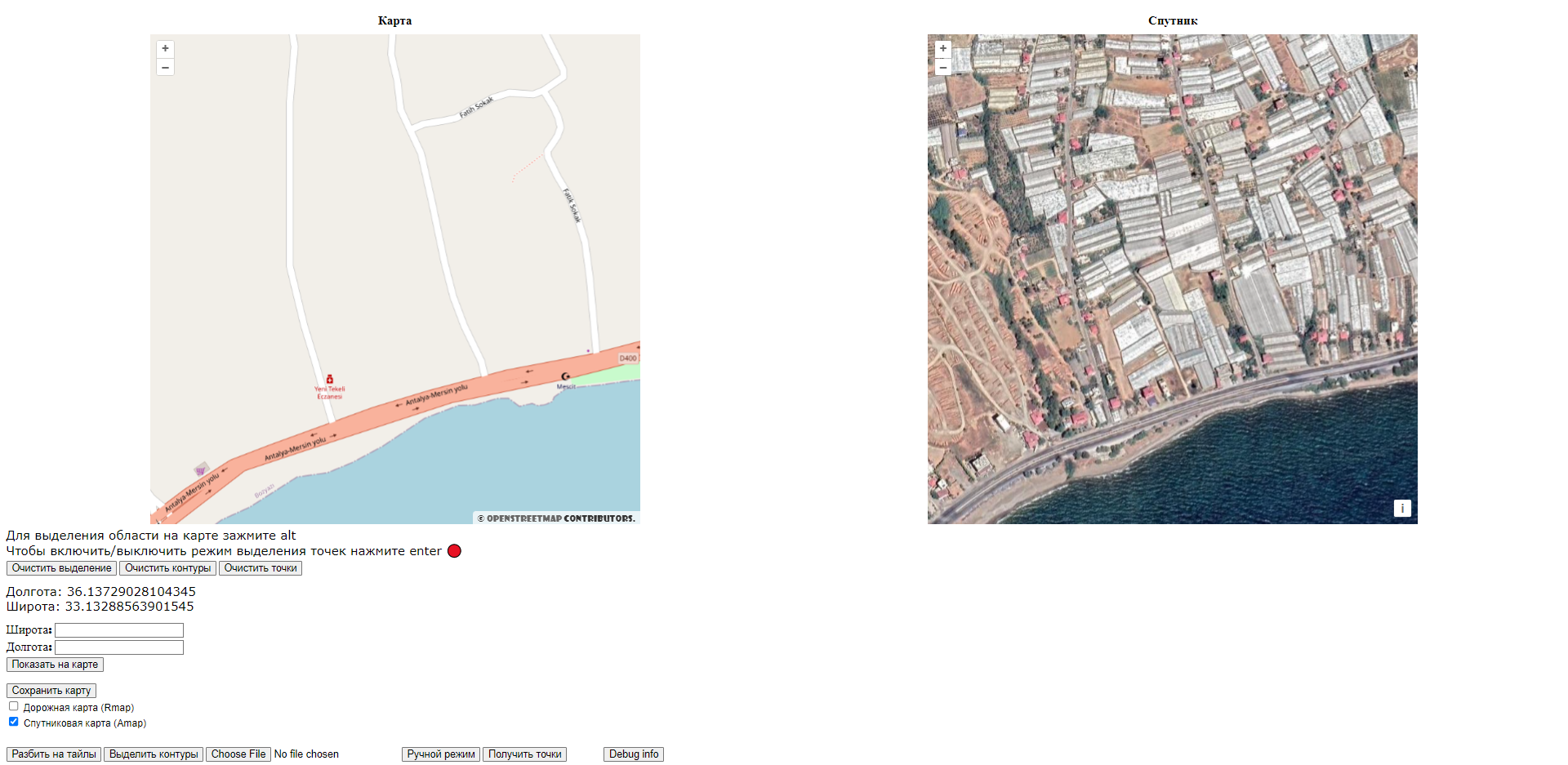
****

Рисунок 1 – Общий вид интерфейса программы

### 2.2.1 Карты

На странице пользователь видит две карты: спутниковая карта и карта дорог. Снизу карт находится панель управления с кнопками и полями ввода.

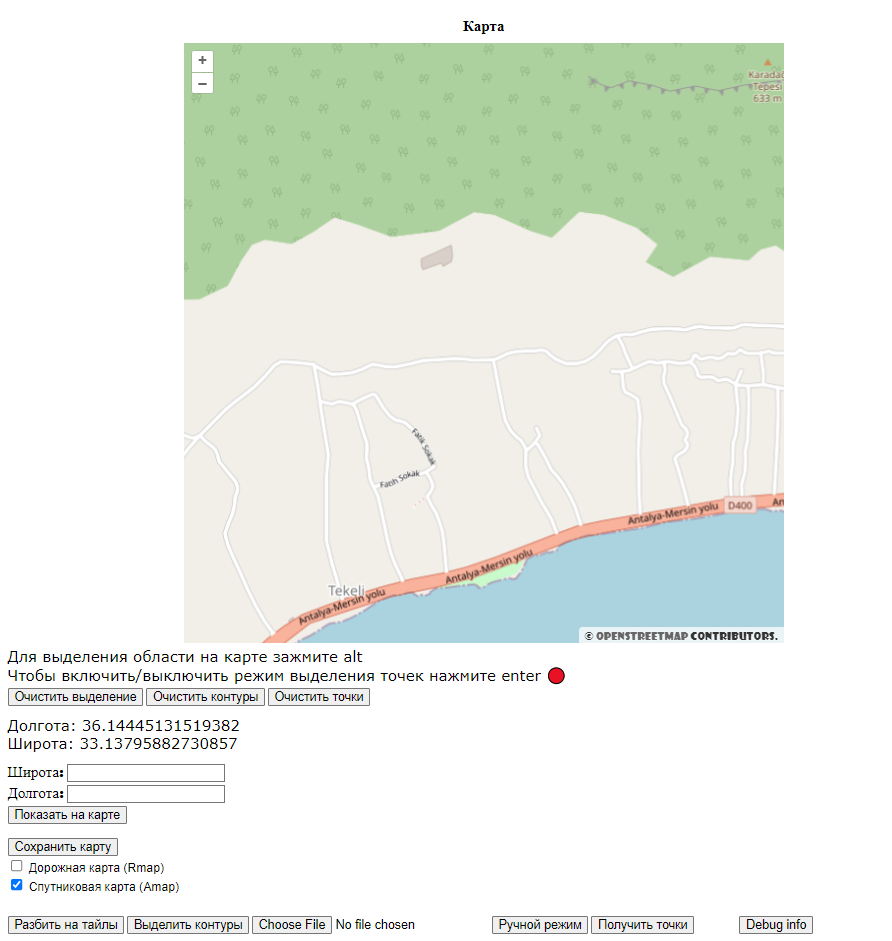
****

Рисунок 2– Карта дорог и панель управления



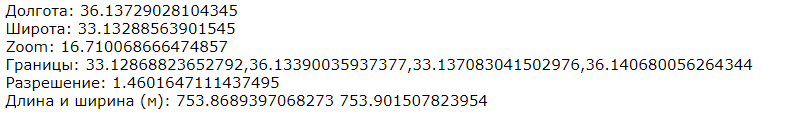
Рисунок 3 – Спутниковая карта

### 2.2.2 Координаты

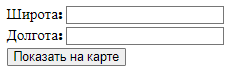
На экран выводятся географически координады (широту и долготу) центра обозреваемого фрагмента карты.



Также пользователь может нажать кнопку "Debug info" для отображения дополнительной информации такой как: zoom(приближение), географические координаты границ наблюдаемого участка (леывй верхний и правый нижний угол), разрешение изображения, длина и ширина наблюдаемого участка в метрах; эта дополнительная информация не совсем нужна обычному пользователю, но нужна для использования другими программистами, например для выделения изображений с точным разрешением для нейронной сети и вычисления площади теплиц по линейным размерам кадра. Повторное нажатие кнопки выключит отображение дополнительной информации.

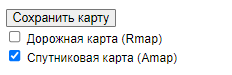


Предусмотрены поля для ввода координат самим пользователем.



### 2.2.3 Сохранение карт

Можно скачать текущее изображение карты, нажав кнопку сохранить карту, при этом все маркеры, такие как области выделения, контуры объектов, и точки, установленные пользователем будут на ней сохранены. В соответствии с установленной галочкой будет скачана карта дорог под названием "Rmap.jpg" и/или спутниковая карта под названием "Amap.jpg".



### 2.2.4 Выделение области поиска теплиц

Для работы программы пользователь должен выбрать область на карте. Это можно сделать, если зажать клавишу alt и провести мышкой по карте; тогда выделиться прямоугольная область которая будет обрабатываться программой. Область можно выбрать на любой из двух карт, и она будет дублироваться на вторую карту. Область выделения можно задать только одну. Очистить область выделения можно кнопкой очистить выделение.

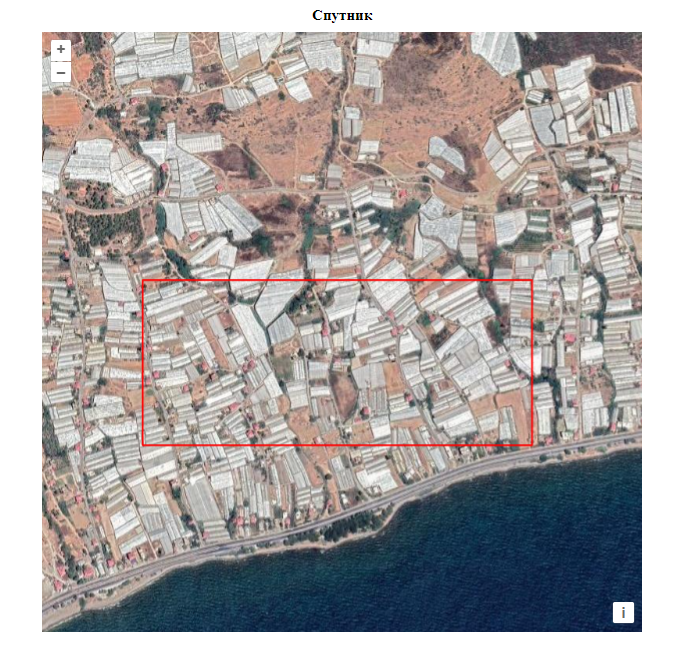


Рисунок 5 – Область выделения на карте

### 2.2.5 Разбиение участка на тайлы.

Пользователь выбирает относительно большой участок земной поверхности, на котором такие маленькие объекты как теплицы невозможно распознать. Поэтому данный участок разбивается на участки с более крупным разрешением, где видны контуры объектов.

При нажатии кнопки "разбить на тайлы" будет скачан массив изображений. Изображения будут подписаны как "0-map.png", "1-map.png" и так далее. При этом пользовательские маркеры (области выделения, контуры объектов, точки) НЕ сохраняются.

Также будет скачан текстовый документ "tile-coordinates.txt", который содержит информацию по заказу программистов, занимающимися распознаванием контуров теплиц.

Первая строка: географическая ширина тайла, географическая длина тайла — они равны для всех тайлов.

Далее идет: номер тайла, широта центра, долгота центра

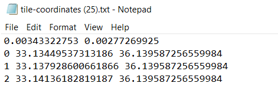


Рисунок 6 – Структура файла tile-coordinates.txt

### 2.2.6 Выделение контуров

Получив разбиение на тайлы участка, можно выделить контуры теплиц. Отобразить контуры можно на карте, с помощью кнопки "отобразить контуры", но для этого нужно предварительно загрузить файл в формате txt с координатами контуров. Этот файл мы получаем от других программистов занимающимися распознаванием контуров. Очистить контуры можно кнопкой "очистить контуры".

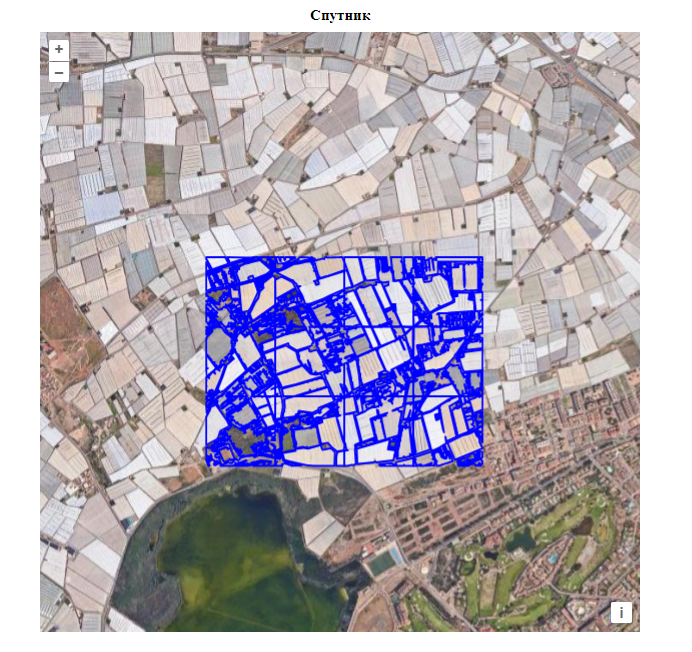


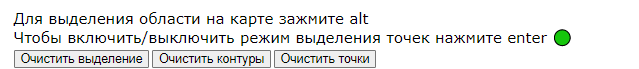
Рисунок 7 – Выделенные контуры на карте

### 2.2.7 Установка точек на карте

Пользователь может установить на карте точки. Чтобы включить/выключить режим установки точек нужно нажать клавишу enter.



Красная точка — режим установки точек выключен



Зеленая точка — режим установки точек включен

При включенном режиме клик по карте установит точку, а координаты точки запишутся в файл "point-coordinates.txt". Файл имеет следующий формат: индекс точки, широта, долгота. Точки, просто выделенные на карте, записываются с индексом -1. Если точка находится внутри тайла из разбиения участка, то ей присваивается индекс равный номеру тайла.

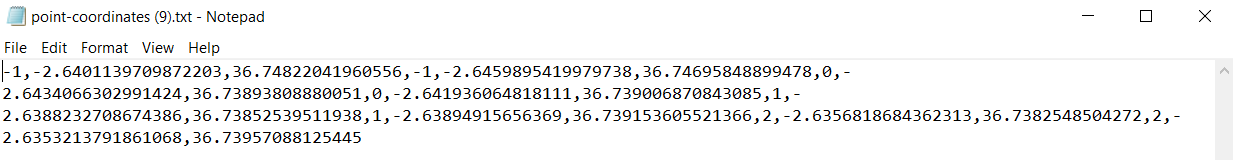


Рисунок 8 – Структура файла point-coordinates.txt

### 2.2.8 Ручной режим

Если нажать кнопку "ручной режим", то перед пользователем будет показан массив изображений, который составляют выделенную область. На этих изображениях пользователь может ставить точки, индекс точки в таком случае будет равен индексу текущего тайла из массива. Этот режим предназначен для определения теплиц человеком; оператор может просто отметить точками теплицы, после чего программисты получат список координат этих теплиц. Также можно предварительно выделить контуры и дать пользователю выбрать точкой контуры теплиц, это нужно для обучения нейронной сети. Очистить точки можно кнопкой "очистить точки".



Рисунок 9 – Ручной режим без выделенных контуров

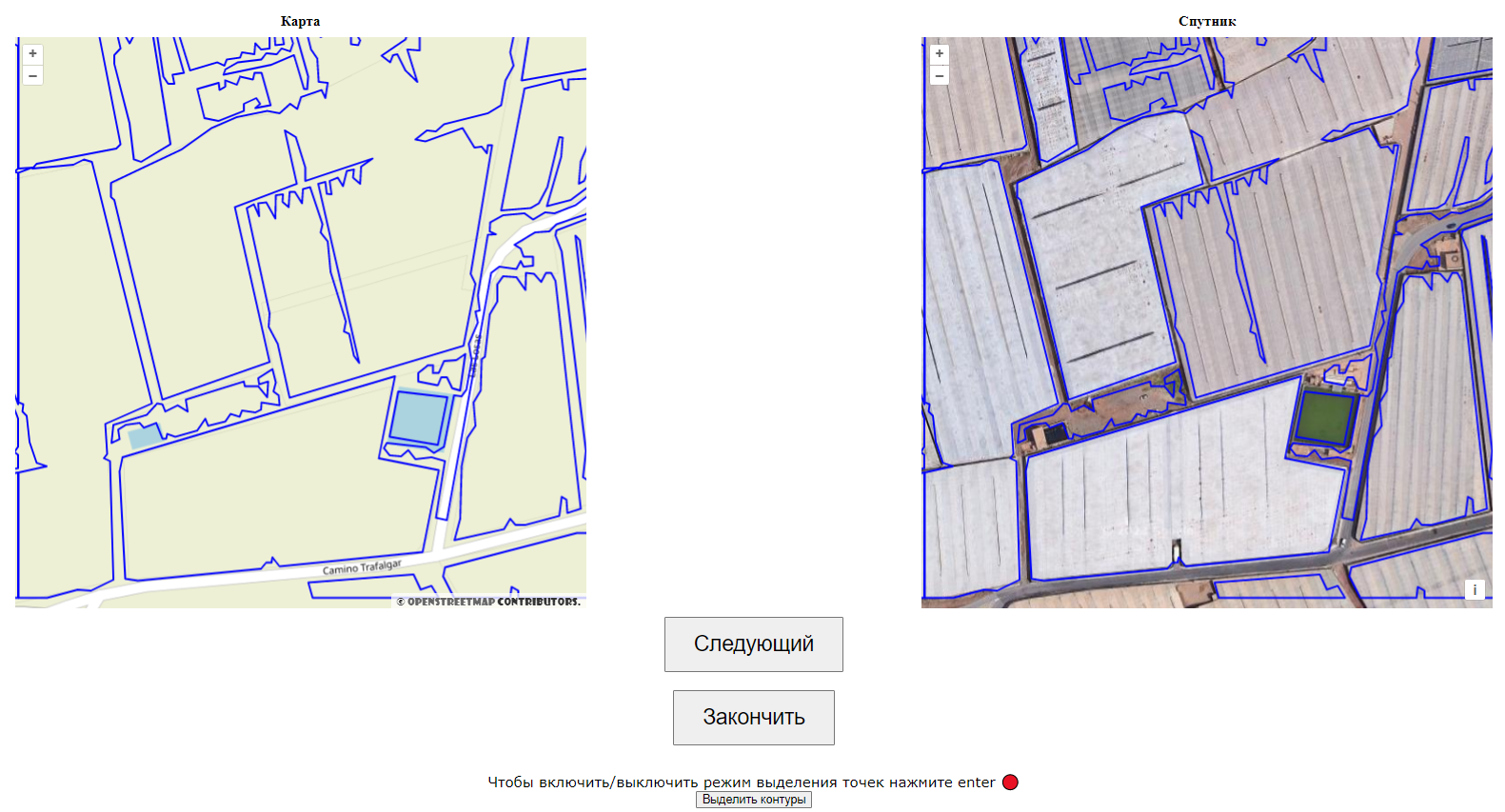


Рисунок 10 – Ручной режим с выделенными контурами

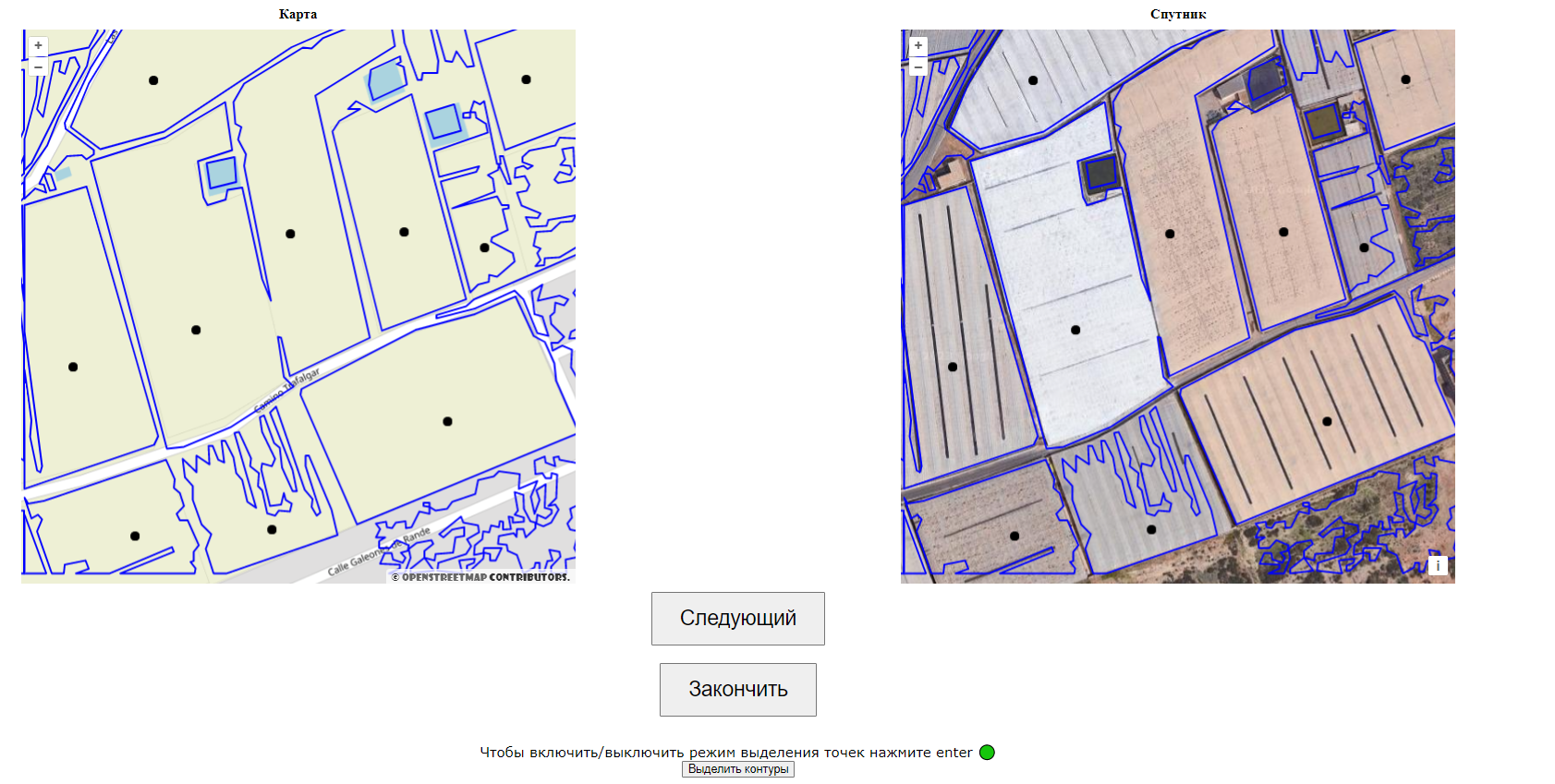


Рисунок 11 – Установленные точки в ручном режиме

# 3. Разработка ПО на основе OpenLayers

Весь код проекта можно найти в репозитории на github по ссылке: <https://github.com/skadi1337/OpenLayers>.

## 3.1. Index.html

Программа представляет собой SPA (single page application) — это веб-приложение, которое загружает только одну HTML-страницу и динамически обновляет ее содержимое в зависимости от действий пользователя. Код ниже представляет собой HTML-страницу, которая содержит интерфейс программы. Вся логика взаимодействия с картой реализована в файле main.js, который подключается в конце страницы. Стили реализованы в отдельном файле styles.css, который подключается в <head> с помощью тега <link>.

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Поисковик теплиц</title>

    <link rel="stylesheet" href="node\_modules/ol/ol.css">

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <div class="wrapper">

        <div class="half">

            <h4>Карта</h4>

            <div id="roadMap" class="map"></div>

        </div>

        <div class="half">

            <h4>Спутник</h4>

            <div id="aerialMap" class="map"></div>

        </div>

    </div>

    <div id="toolsDiv">

        <div>

            <p>Для выделения области на карте зажмите alt</p>

            <p id="pointSelectHint">Чтобы включить/выключить режим выделения точек нажмите enter 🔴</p>

            <button id="clearSelectionButton">Очистить выделение</button>

            <button id="clearConturButton">Очистить контуры</button>

            <button id="clearPoints">Очистить точки</button>

            <br>

        </div>

        <div id="coordinates"></div>

        <div>

            <form>

                <label for="latitude">Долгота:</label>

                <input type="text" id="latitude" name="latitude">

                <br>

                <label for="longitude">Широта:</label>

                <input type="text" id="longitude" name="longitude">

                <br>

                <button type="button" id="showOnMapButton">Показать на карте</button>

            </form>

        </div>

        <div>

            <button id="saveMapButton">Сохранить карту</button>

            <br>

            <label for="checkbox1" class="checkbox-label">

                <input type="checkbox" id="checkbox1" name="checkbox1"> Дорожная карта (Rmap)

            </label>

            <br>

            <label for="checkbox2" class="checkbox-label">

                <input type="checkbox" id="checkbox2" name="checkbox2" checked> Спутниковая карта (Amap)

            </label>

            <a id="RmapDownload" download="Rmap.png"></a>

            <a id="AmapDownload" download="Amap.png"></a>

        </div>

        <br>

        <div id="tileButtons">

            <button id="makeTilesButton">Разбить на тайлы</button>

            <button id="drawContursButton">Выделить контуры</button>

            <input type="file" id="file-input">

            <button id="makeHandSelection">Ручной режим</button>

            <button id="getPoints">Получить точки</button>

            <button id="switchDebug">Debug info</button>

        </div>

    </div>

    <div id="handSelectDiv" displat="none">

        <button id="nextTile">Следующий</button>

        <button id="endHandSelect">Закончить</button>

        <br>

        <p id="pointSelectHint2">Чтобы включить/выключить режим выделения точек нажмите enter 🔴</p>

        <button id="showContursInHandSelect">Выделить контуры</button>

    </div>

    <script src="https://cdnjs.cloudflare.com/ajax/libs/FileSaver.js/2.0.5/FileSaver.min.js"></script>

    <script src="https://cdn.jsdelivr.net/npm/elm-pep@1.0.6/dist/elm-pep.js"></script>

    <script type="module" src="main.js"></script>

</body>

</html>

## 3.2. Main.js

В этом файле реализована основная логика программы. Ниже приведена основная часть кода, т.к. файл слишком большой, чтобы вставить его полностью. Код оформлен с комментариями, поясняющими предназначения того или иного фрагмента кода.

import BingMaps from 'ol/source/BingMaps.js';

import Map from 'ol/Map.js';

import TileLayer from 'ol/layer/Tile.js';

import View from 'ol/View.js';

import OSM from 'ol/source/OSM';

import { Vector as VectorLayer } from 'ol/layer'

import { Vector } from 'ol/source'

import \* as olSphere from 'ol/sphere';

import { Draw, DragBox } from 'ol/interaction'

import { altKeyOnly } from 'ol/events/condition'

import { Polygon } from 'ol/geom'

import Feature from 'ol/Feature';

import { Style, Stroke, Fill, Circle } from 'ol/style'

import { boundingExtent, getTopLeft, getTopRight, getBottomRight, getBottomLeft } from 'ol/extent'

import { fromLonLat, transform, useGeographic, toLonLat, disableCoordinateWarning } from 'ol/proj.js';

import XYZ from 'ol/source/XYZ';

const TARGET\_ZOOM = 18              // Значения zoom для разбиения на тайлы

const LAT\_DELTA\_400 = 0.00214576721 // для zoom = 18, и 400х400

const LON\_DELTA\_400 = 0.00173286842 // если изменить параметры, то измениться и дельта

const LAT\_DELTA = 0.00343322753     //18 zoom, 640x640

const LON\_DELTA = 0.00277269925

const LAT\_DELTA\_METERS = 308.298566865291   // Линейный размеры кадра в метрах, при зуме 18

const LON\_DELTA\_METERS = 308.3040143497024;

const MAX\_TILES = 100           // макс кол-во тайлов в разбиении

const base\_extent = [-180, -85, 180, 85] // Базовые границы области выдимости (весь земной шар)

var current\_zoom;

var current\_center;

var is\_debug = false;

var selected\_extent = null;

var isDrawingPoints = false;

var curr\_tile = -1;

var poinstArray = [];

var pointSelectHint = document.getElementById('pointSelectHint');

var pointSelectHint2 = document.getElementById('pointSelectHint2');

var showConturs = false;

var curr\_x, curr\_y = null, curr\_final\_x, curr\_final\_y, curr\_top\_left, curr\_bot\_right;

// Необходимо для перехода OpenLayers на географические координаты

useGeographic();

// Включение/Выключение режима выделения точек

document.addEventListener('keydown', function (evt) {

    if (evt.keyCode === 13) {

        isDrawingPoints = !isDrawingPoints;

    }

    if (isDrawingPoints) {

        pointSelectHint.innerHTML = 'Чтобы включить/выключить режим выделения точек нажмите enter 🟢'

        pointSelectHint2.innerHTML = 'Чтобы включить/выключить режим выделения точек нажмите enter 🟢'

        map1.addInteraction(drawPoint1);

        map2.addInteraction(drawPoint2);

    }

    else {

        pointSelectHint.innerHTML = 'Чтобы включить/выключить режим выделения точек нажмите enter 🔴'

        pointSelectHint2.innerHTML = 'Чтобы включить/выключить режим выделения точек нажмите enter 🔴'

        map1.removeInteraction(drawPoint1);

        map2.removeInteraction(drawPoint2);

    }

});

// Источник спутниковой карты, взят из google maps

const googleSatelliteSource = new XYZ({

    url: 'http://mt1.google.com/vt/lyrs=s&x={x}&y={y}&z={z}', // http://mt0.google.com/vt/lyrs=y&hl=en&x={x}&y={y}&z={z}&s=Ga

    attributions: [

        '© Google'

    ],

    crossOrigin: 'Anonymous'

});

// Слой со спутниковой картой

const googleSatelliteLayer = new TileLayer({

    source: googleSatelliteSource

});

// Слой с картой дорог, источник взят из сервисв OpenStreetMaps (OSM)

const roadLayer = new TileLayer({

    source: new OSM(),

});

// слой с областью выделения

var highlightLayer = new VectorLayer({

    source: new Vector(),

    style: new Style({

        stroke: new Stroke({

            color: 'red',

            width: 2

        }),

        fill: new Fill({

            color: 'rgba(255, 0, 0, 0)'

        })

    })

});

// слой с контурами

var highlightLayer2 = new VectorLayer({

    source: new Vector(),

    style: new Style({

        stroke: new Stroke({

            color: 'blue',

            width: 2

        }),

        fill: new Fill({

            color: 'rgba(0, 255, 0, 0)'

        })

    })

});

// слой с точками

var pointsLayer = new VectorLayer(

    {

        source: new Vector(),

        style: new Style({

            image: new Circle({

                radius: 5,

                fill: new Fill({

                    color: 'black'

                }),

                stroke: new Stroke({

                    color: 'black',

                    width: 1

                })

            })

        })

    }

)

// Это view, он же видимы кадр. Для обеих карт используется один view.

const view = new View({

    center: [33.13329651842474, 36.137167149366945],

    zoom: 10,

    extent: base\_extent

});

// Карта дорог

const map1 = new Map({

    target: 'roadMap',

    layers: [roadLayer],

    view: view,

});

// Карта спутника

const map2 = new Map({

    target: 'aerialMap',

    layers: [googleSatelliteLayer],

    view: view,

});

// Области выделения

var selection1 = new DragBox({

    condition: altKeyOnly,

});

// Выделение точки

var drawPoint1 = new Draw({

    source: pointsLayer.getSource(),

    type: 'Point',

    style: new Style({

        image: new Circle({

            radius: 0,

            fill: null,

            stroke: null

        })

    })

});

// Добавление точки в массив при ее установке на карте

drawPoint1.on('drawend', function (event) {

    var feature = event.feature;

    var coords = feature.getGeometry().getCoordinates();

    poinstArray.push(curr\_tile);

    poinstArray.push(coords[0]);

    poinstArray.push(coords[1]);

});

// Обработка события выделения области на карте

selection1.on('boxend', function (event) {

    var geometry = selection1.getGeometry();

    var extent = geometry.getExtent();

    var bottomLeft = toLonLat(getBottomLeft(extent));

    var topRight = toLonLat(getTopRight(extent));

    selected\_extent = []

    selected\_extent.push(bottomLeft[0]);

    selected\_extent.push(bottomLeft[1]);

    selected\_extent.push(topRight[0]);

    selected\_extent.push(topRight[1]);

    var feature = new Feature({

        geometry: new Polygon([

            [

                [bottomLeft[0], bottomLeft[1]],

                [bottomLeft[0], topRight[1]],

                [topRight[0], topRight[1]],

                [topRight[0], bottomLeft[1]],

                [bottomLeft[0], bottomLeft[1]]

            ]

        ])

    });

    highlightLayer.getSource().clear();

    highlightLayer.getSource().addFeature(feature);

});

selection2.on('boxend', function (event) {

    var geometry = selection2.getGeometry();

    var extent = geometry.getExtent();

    var bottomLeft = toLonLat(getBottomLeft(extent));

    var topRight = toLonLat(getTopRight(extent));

    selected\_extent = []

    selected\_extent.push(bottomLeft[0]);

    selected\_extent.push(bottomLeft[1]);

    selected\_extent.push(topRight[0]);

    selected\_extent.push(topRight[1]);

    var feature = new Feature({

        geometry: new Polygon([

            [

                [bottomLeft[0], bottomLeft[1]],

                [bottomLeft[0], topRight[1]],

                [topRight[0], topRight[1]],

                [topRight[0], bottomLeft[1]],

                [bottomLeft[0], bottomLeft[1]]

            ]

        ])

    });

    highlightLayer.getSource().clear();

    highlightLayer.getSource().addFeature(feature);

});

// Отрисовка информации о карте (широта, долгота + debug info)

function showCoordinates() {

    var center = view.getCenter();

    var zoom = view.getZoom();

    var extent = view.calculateExtent()

    var res = view.getResolution()

    // bot left

    var p1 = []

    p1.push(extent[0])

    p1.push(extent[1])

    //top left

    var p2 = []

    p2.push(extent[0])

    p2.push(extent[3])

    //bot right

    var p3 = []

    p3.push(extent[2])

    p3.push(extent[1])

    var xd = olSphere.getDistance(p1, p2);

    var yd = olSphere.getDistance(p1, p3);

    var coordinates = document.getElementById('coordinates');

    coordinates.innerHTML = 'Долгота: ' + center[1] + '<br>Широта: ' + center[0];

    if (is\_debug) {

        coordinates.innerHTML = coordinates.innerHTML + '<br>\nZoom: ' + zoom + '<br>Границы: ' + extent + '<br>Разрешение: ' + res

            + '<br>Длина и ширина (м): ' + String(xd) + ' ' + String(yd);

    }

}

// Функция сохранения тайла при разбиении учатска на тайлы

function take\_screen(x, y, i) {

    const mapCanvas = document.createElement('canvas');

    const size = map2.getSize();

    mapCanvas.width = size[0];

    mapCanvas.height = size[1];

    const mapContext = mapCanvas.getContext('2d');

    Array.prototype.forEach.call(

        map2.getViewport().querySelectorAll('.ol-layer canvas, canvas.ol-layer'),

        function (canvas) {

            if (canvas.width > 0) {

                const opacity =

                    canvas.parentNode.style.opacity || canvas.style.opacity;

                mapContext.globalAlpha = opacity === '' ? 1 : Number(opacity);

                let matrix;

                const transform = canvas.style.transform;

                if (transform) {

                    matrix = transform

                        .match(/^matrix\(([^\(]\*)\)$/)[1]

                        .split(',')

                        .map(Number);

                } else {

                    matrix = [

                        parseFloat(canvas.style.width) / canvas.width,

                        0,

                        0,

                        parseFloat(canvas.style.height) / canvas.height,

                        0,

                        0,

                    ];

                }

                CanvasRenderingContext2D.prototype.setTransform.apply(

                    mapContext,

                    matrix

                );

                const backgroundColor = canvas.parentNode.style.backgroundColor;

                if (backgroundColor) {

                    mapContext.fillStyle = backgroundColor;

                    mapContext.fillRect(0, 0, canvas.width, canvas.height);

                }

                mapContext.drawImage(canvas, 0, 0);

            }

        }

    );

    mapContext.globalAlpha = 1;

    mapContext.setTransform(1, 0, 0, 1, 0, 0);

    const link = document.createElement('a');

    link.href = mapCanvas.toDataURL();

    link.download = String(i) + '-map.png';

    document.body.appendChild(link);

    link.click();

    document.body.removeChild(link);

    map2.renderSync();

}

// Функция разбиения области выделения на тайлы

async function tileUp() {

    if (selected\_extent === null) {

        alert("Область не выбрана");

        return;

    }

    var extent = selected\_extent;

    highlightLayer.setVisible(false);

    pointsLayer.setVisible(false);

    var top\_left = [extent[0], extent[1]];

    var bot\_right = [extent[2], extent[3]];

    var base\_center = view.getCenter();

    var base\_zoom = view.getZoom();

    var xdelta = LAT\_DELTA / 2;

    var ydelta = LON\_DELTA / 2;

    var x = top\_left[0] + xdelta;

    var y = top\_left[1] + ydelta;

    var final\_x = bot\_right[0];

    var final\_y = bot\_right[1];

    var xcnt = (final\_x - x) / LAT\_DELTA;

    var ycnt = (final\_y - y) / LON\_DELTA;

    if (xcnt + ycnt > MAX\_TILES) {

        alert("Слишком маленький масштаб");

        return;

    }

    var i = 0;

    var text = String(LAT\_DELTA) + ' ' + String(LON\_DELTA)

    view.setZoom(TARGET\_ZOOM)

    while (y <= final\_y) {

        while (x <= final\_x) {

            view.setCenter([x, y]);

            await new Promise((resolve) => {

                map2.once('rendercomplete', resolve);

            });

            text = text + '\n' + String(i) + ' ' + String(x) + ' ' + String(y)

            take\_screen(x, y, i)

            x = x + LAT\_DELTA;

            i = i + 1;

        }

        y = y + LON\_DELTA;

        x = top\_left[0] + xdelta;

    }

    var filename = "tile-coordinates.txt";

    var blob = new Blob([text], { type: "text/plain;charset=utf-8" });

    saveAs(blob, filename);

    view.setCenter(base\_center);

    view.setZoom(base\_zoom);

    highlightLayer.setVisible(true);

    pointsLayer.setVisible(true);

}

const toolsDiv = document.getElementById("toolsDiv");

const handSelectDiv = document.getElementById("handSelectDiv");

// Функция "ручного режима"

function handSelect() {

    if (selected\_extent === null) {

        alert("Область не выбрана");

        return;

    }

    current\_center = view.getCenter();

    current\_zoom = view.getZoom();

    toolsDiv.style.display = "none";

    handSelectDiv.style.display = "flex";

    var extent = selected\_extent;

    highlightLayer.setVisible(false);

    var top\_left = [extent[0], extent[1]];

    var bot\_right = [extent[2], extent[3]];

    curr\_top\_left = top\_left;

    curr\_bot\_right = bot\_right;

    var xdelta = LAT\_DELTA / 2;

    var ydelta = LON\_DELTA / 2;

    var x = top\_left[0] + xdelta;

    var y = top\_left[1] + ydelta;

    curr\_x = x;

    curr\_y = y;

    var final\_x = bot\_right[0];

    var final\_y = bot\_right[1];

    curr\_final\_x = final\_x;

    curr\_final\_y = final\_y;

    var xcnt = (final\_x - x) / LAT\_DELTA;

    var ycnt = (final\_y - y) / LON\_DELTA;

    if (xcnt + ycnt > MAX\_TILES) {

        alert("Слишком маленький масштаб");

        return;

    }

    var i = 0;

    curr\_tile = 0;

    view.setZoom(TARGET\_ZOOM)

    view.setCenter([x, y]);

}

// Функция загрузки файла с координатами контуров

function parseFile()

{

    var fileInput = document.getElementById('file-input');

    var file = fileInput.files[0];

    if (!file) {

        alert('Пожалуйста, выберете файл с контурами.');

        return;

    }

    showConturs = true;

    var fileReader = new FileReader();

    fileReader.onload = function (event) {

        var fileContent = event.target.result;

        var lines = fileContent.trim().split('\n');

        lines.forEach(function (line) {

            var numbers = [];

            var lineNumbers = line.trim().split(/\s+/);

            lineNumbers.forEach(function (number) {

                numbers.push(parseFloat(number));

            });

            numbers.push(numbers[0]);

            numbers.push(numbers[1]);

            var pairs = [];

            for (var i = 0; i < numbers.length; i += 2) {

                var pair = [numbers[i], numbers[i + 1]];

                pairs.push(pair);

            }

            var coordinates = [pairs];

            var feature = new Feature({

                geometry: new Polygon(coordinates)

            });

            highlightLayer2.getSource().addFeature(feature)

        });

    };

    fileReader.readAsText(file);

}

// Функция для скачивания файла с координатами точек

function downloadPointsFile() {

    var len = poinstArray.length;

    if (len == 0) {

        alert('Не выбрано ни одной точки.');

        return;

    }

    var text = String(poinstArray);

    var filename = "point-coordinates.txt";

    var blob = new Blob([text], { type: "text/plain;charset=utf-8" });

    saveAs(blob, filename);

}

// Функция перехода на следующий тайл в ручном режиме

function nextTile() {

    if (curr\_y == null) {

        return;

    }

    if (curr\_x > curr\_final\_x) {

        curr\_y = curr\_y + LON\_DELTA;

        curr\_x = curr\_top\_left[0] + (LAT\_DELTA / 2);

        if (curr\_y > curr\_final\_y) {

            alert('Изображения закончились');

            endHandSelect();

            return;

        }

        nextTile();

    }

    else {

        curr\_tile = curr\_tile + 1;

        curr\_x = curr\_x + LAT\_DELTA;

        view.setCenter([curr\_x, curr\_y]);

    }

}

// Функция заверщения ручного режима

function endHandSelect() {

    toolsDiv.style.display = "block";

    handSelectDiv.style.display = "none";

    view.setCenter(current\_center);

    view.setZoom(current\_zoom);

    curr\_tile = -1;

    curr\_y = null;

    highlightLayer2.setVisible(true);

    highlightLayer.setVisible(true);

}

function showContursInHandSelect()

{

    if (showConturs == false)

    {

        highlightLayer2.setVisible(true);

    }

    else

    {

        highlightLayer2.setVisible(false);

    }

    showConturs = !showConturs;

}

const checkbox1 = document.getElementById('checkbox1');

const checkbox2 = document.getElementById('checkbox2');

// Далее идет привязка кнопок из index.html к функциям выше

var button = document.getElementById('showOnMapButton');

button.addEventListener('click', updateMap);

var buttonTiles = document.getElementById('makeTilesButton');

buttonTiles.addEventListener('click', tileUp)

var buttonHandSelect = document.getElementById('makeHandSelection');

buttonHandSelect.addEventListener('click', handSelect)

var clearConturButton = document.getElementById('clearConturButton');

clearConturButton.addEventListener('click', clearConturs)

var clearPointsButton = document.getElementById('clearPoints');

clearPointsButton.addEventListener('click', clearPoints);

var clearSelection = document.getElementById('clearSelectionButton');

clearSelection.addEventListener('click', clearSelectes)

var drawConturs = document.getElementById('drawContursButton');

drawConturs.addEventListener('click', parseFile)

var switchDebug = document.getElementById('switchDebug');

switchDebug.addEventListener('click', function () { is\_debug = !is\_debug; showCoordinates() })

var getPointsButton = document.getElementById('getPoints');

getPointsButton.addEventListener('click', downloadPointsFile)

var nextTileButton = document.getElementById('nextTile');

nextTileButton.addEventListener('click', nextTile);

var endHandSelectButton = document.getElementById('endHandSelect');

endHandSelectButton.addEventListener('click', endHandSelect)

var showContursInHandSelectButton = document.getElementById('showContursInHandSelect');

showContursInHandSelectButton.addEventListener('click', showContursInHandSelect)

## 3.3 Styles.css

@viewport {

  zoom: 0.75;

}

.wrapper {

  display: flex;

  justify-content: center;

  align-items: center;

  margin-top: 0;

}

.half {

  width: 50%;

  padding: 0 10px;

  box-sizing: border-box;

  text-align: center;

}

.map {

  width: 640px;

  height: 640px;

  margin: 0 auto;

}

h4 {

  margin-top: 0.5em;

  margin-bottom: 0.5em;

}

#map-titles {

  margin-top: 0;

}

#coordinates {

  font-family: Verdana, sans-serif;

}

form {

  margin-top: 10px;

}

#saveMapButton {

  margin-top: 15px;

}

#coordinates {

  margin-top: 10px;

}

#getPoints {

  margin-left: 0 auto;

}

#switchDebug {

  margin-left: 45px;

}

#handSelectDiv {

  display: none;

  flex-direction: column;

  align-items: center;

  justify-content: center;

  height: 100%;

}

#nextTile,

#endHandSelect {

  font-size: 24px;

  padding: 16px 32px;

  margin: 10px;

}

a#RmapDownload,

a#AmapDownload {

  display: none;

}

#toolsDiv {

  margin-top: 5px;

}

#filePath {

  resize: none;

  overflow: hidden;

  height: 30px;

  width: 100%;

  padding: 5px;

  font-size: 16px;

  border: 1px solid #ccc;

  border-radius: 5px;

}

input[type="checkbox"] {

  width: 12px;

  height: 12px;

  border-radius: 50%;

  border: 1px solid #000;

  outline: none;

}

input[type="checkbox"]:checked {

  background-color: #000;

}

label.checkbox-label {

  font-family: Arial, sans-serif;

  font-size: 13px;

}

p {

  margin: 0;

  font-size: 16px;

  font-family: Verdana, sans-serif;

}

# ЗАКЛЮЧЕНИЕ

В результате выполнения данной курсовой работы была разработана система анализа данных дистанционного зондирования на основе технологии OpenLayers.

В процессе работы были рассмотрены основные принципы дистанционного зондирования и технические характеристики данных, а также особенности использования технологии OpenLayers для создания интерактивных карт и визуализации геоданных с высокой степенью точности и качества. Были проведены исследования по использованию получаемых данных для распознавания образов и обучения нейронных сетей.

Были определены оптимальные настройки системы, включая выбор проекции карты, настройку взаимодействия пользователей с картой и механизмов визуализации данных. Эти настройки позволяют использовать систему анализа данных дистанционного зондирования с максимальной эффективностью и удобством для пользователей и других программистов.

Результаты подтверждают возможность успешного использования технологии OpenLayers для создания систем анализа данных дистанционного зондирования, что открывает новые возможности для анализа и визуализации геоданных в различных областях, включая экологию, геологию, агрономию и многие другие.

На основе проведенной работы, можно рекомендовать ее использование в различных сферах, связанных с обработкой и анализом геоданных.

В дальнейшем, возможно, будет целесообразно расширить функциональность системы, добавив новые инструменты анализа и визуализации данных, а также улучшить механизмы обработки больших объемов данных, что позволит повысить эффективность работы системы.

Также необходимо отметить, что использование технологии OpenLayers требует определенных знаний и навыков в области программирования и геоинформатики, поэтому для успешного использования системы необходимо обеспечить подготовку и обучение пользователей.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Флэнаган, "JavaScript. Полное руководство", 7-е издание [2021].
2. Dr. Axel Rauschmayer, "Speaking JavaScript" [Электронный ресурс] // <https://exploringjs.com/es5/toc.html> — 02.04.2023
3. MDN Web Docs [Электронный ресурс] // <https://developer.mozilla.org/ru/> — 02.04.2023
4. OpenLayers.org [Электронный ресурс] // <https://openlayers.org/en/latest/examples/> — 02.04.2023