Interactive Coastal Erosion Map with AI Predictions

Authors

Primary Author: Theophile Avenel

Supervisor: Bahareh Kamranzad

Contributors: Hachem Kassem

Abstract

This technical paper explores the development of an interactive map to visualize coastal erosion using AI predictions. By integrating data from satellite imagery and other sources, we aim to provide a comprehensive tool for understanding shoreline changes over time. The project uses technologies such as Mapbox for visualization and various AI techniques for predictive modeling. In addition, a state-of-the-art review will be conducted to discuss existing tools and methodologies for coastal erosion visualization and prediction, highlighting current advancements and identifying gaps in the research.

Introduction

Many research articles focus on shoreline extraction using satellite imagery, but few address the creation of an interactive map to visualize these changes. This paper aims to bridge that gap by developing an interactive map that uses AI predictions to visualize coastal erosion.

Please note that the website/blog for my research topic is available at https://erosion-ai.tavenel.fr/

I fand two projects which created an interactive map: the CoastSat project, detailed on this site and described in this scientific article, and the Aqua Monitor , which shows how the Earth's surface water has changed over the last 30 years, available here and explained in this Nature Climate Change paper.

I haven't seen any articles that discuss how to represent coastal erosion on an interactive map because the shoreline is always changing; however, the coastline does not alter significantly.

Observations and Definition of the Coastline

Coastal erosion visualization on interactive maps presents a unique challenge due to the dynamic nature of shorelines. As shorelines shift frequently, determining the precise location of the "coastline" becomes complex. The definition of a coastline can vary based on different criteria, such as tidal ranges or sediment movements.

The image (Figure 1) below illustrates different representations of coastlines using various data sources. Each colored line represents a distinct method of defining the coastline:

- **OpenStreetMap Coastline (Blue)**: This line is derived from OpenStreetMap (OSM) data, which is crowd-sourced and might not always be precise. It shows a general outline of the coast based on contributors' input.
- **CoastSat Shoreline (Yellow):** The yellow line represents the shoreline data extracted using the CoastSat algorithm. This method relies on satellite imagery to determine the position of the shoreline with high precision, often identifying recent water levels and sediment deposits.

- **Shoreline Transect (Red):** Transects are specific cross-sections used to measure changes in the shoreline over time. The red lines in the image indicate these transects, which help in analyzing coastal erosion trends and patterns.
- **Real Coastline? (Black):** This hypothetical line suggests what might be considered the "real" coastline based on long-term observations and historical data. It is often used as a reference to compare with other data sources.

- **Various Sand Colors:** Different shades of sand color indicate variations in sediment composition and recent depositional patterns, which also affect the perceived location of the coastline.

- **Recent Water Levels:** The area marked by the yellow shading indicates the extent of recent water levels, showing how far inland the water has reached recently.
- **Rocks:** The presence of rocks in the coastal area can influence the shape and stability of the shoreline, acting as natural barriers against erosion.

Challenges in Defining the Coastline

The main challenge in representing coastal erosion on an interactive map lies in deciding which line to use as the "true" coastline. If you define the coastline as 100 meters from the latest shoreline at the highest tide, the coastline could appear constant for a while. However, this definition might not capture the dynamic changes occurring due to tidal influences, sediment movements, and human activities.

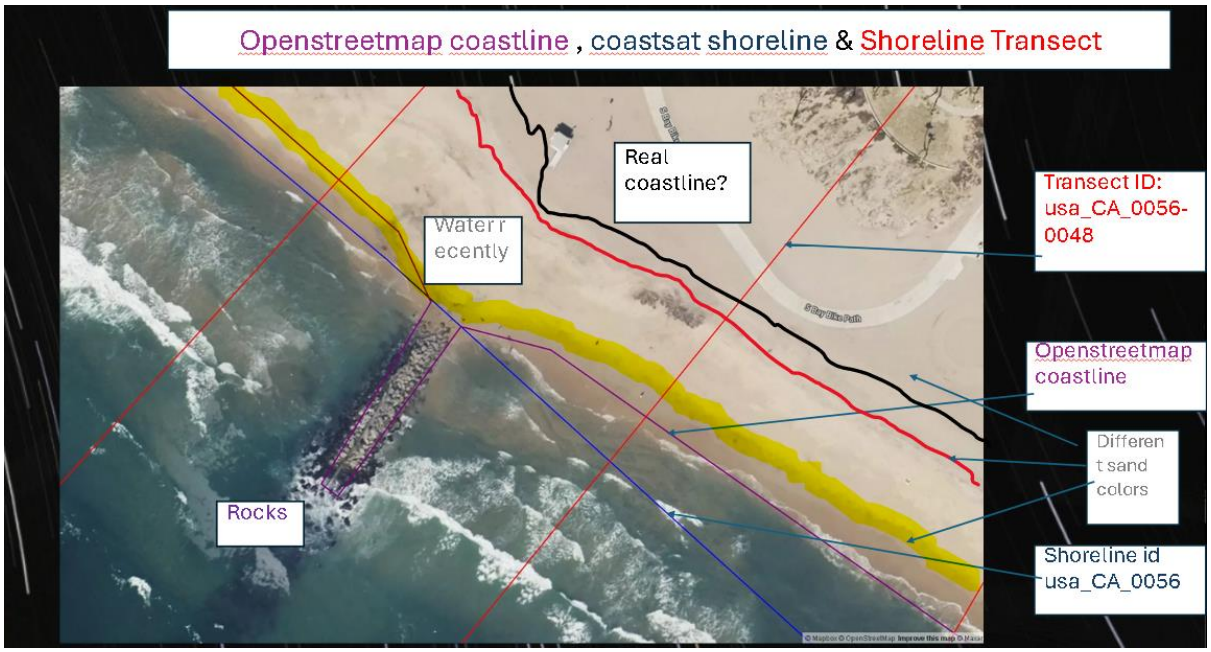

Fig1. Example of different coastline representations using various data sources.

This figure showcases the complexity of defining the coastline and highlights the need for a consistent and precise method to visualize coastal erosion accurately.

The main challenge is how to represent coastal erosion on an interactive map.

Firstly, let's start with a definition of my research topic to better understand the topic.

Definition of My Research Topic

What is an interactive map?

An interactive map is a digital map that allows users to actively engage with the displayed content, providing a dynamic and immersive experience. Unlike static maps, interactive maps offer eatures such as zooming, panning, and clickable elements that provide additional information through pop-ups, detailed descriptions, or hyperlinks.

The Mapbox integration in our project is used to generate an interactive map that unveils coastal dynamics on sandy shores. This can be done by allowing users to freely explore different locations and zoom in and out to overlay historical and predictive images, helping to better understand coastal erosion patterns. Such interactivity makes data analysis more accessible not only for researchers, policymakers, and interested residents but also for a wider audience.

What is coastal erosion?

Coastal erosion is the wearing away of coastlines through natural forces such as waves, tides, currents, and wind. It leads to the loss of land, with shorelines moving back, creating major environmental, economic, and social problems. Waves are a primary cause, constantly battering the shore and removing sand, rocks, and soil. Tidal forces worsen erosion by

continuously altering sea levels, particularly in areas with high tidal ranges. Ocean currents transport sediment away from the coast, depleting beach materials. Human activities such as construction, urbanization, and dredging can disrupt littoral drifts and accelerate erosion.

The effects of coastal erosion are wide and varied, including loss of wildlife habitats, damage to property and infrastructure, and increased susceptibility to storm surges and flooding. Mitigation techniques such as beach nourishment, construction of protective barriers, and sustainable coastal management practices are used to address coastal erosion. However, efforts to address coastal erosion are challenging and require balancing human activity needs along coastlines with the conservation of natural coastal ecosystems.

What are AI predictions in coastal erosion?

The goal of AI predictions in coastal erosion is to forecast past and future trends of coastal erosion based on existing data points.

1. Implementing the OSM Coastline Feature

OpenStreetMap (OSM) has a feature called OSM coastline, which contains data points to create a line representing the coastline. This feature can be explored in more detail at OSM Coastline.

To retrieve JSON data from this coastline feature, I used the Overpass API OSM query, as documented in the Overpass API documentation.

Fig2. Example of the UI interface usage for retrieving coastline data points from OSM using the Overpass API.

Explanation of the Overpass Query

The code snippet provided in the image demonstrates a simple Overpass query designed to fetch coastline data:

/*

This is an example Overpass query.

Try it out by pressing the Run button above!

You can find more examples with the Load tool.

*/

[out:json];

(

way["natural"="coastline"](34.0, -118.5, 34.1, -118.4); // Boundary limit: (south, west, north, east)

);

```
out body;
>;
out skel qt;
```

Here's a brief explanation of each line of the query:

- **[out:json];** Specifies that the output should be in JSON format.
- **(** Begins a group of statements.
- **way["natural"="coastline"](34.0, -118.5, 34.1, -118.4);** Retrieves all ways tagged with natural=coastline within the specified boundary box (south, west, north, east).
- **);** Closes the group of statements.
- **out body;** Outputs the main content of the selected elements in JSON format.
- **>;** Recursively retrieves all nodes and ways referenced by the initial set of elements.
- **out skel qt;** Outputs a skeletal representation of the data, providing a quick and compact summary.

Usage of the Overpass Turbo Interface

The Overpass Turbo interface simplifies running these queries and visualizing the results. It allows users to input custom queries, run them, and see the output directly.

I obtained the following result:

```
{
  "version": 0.6,
  "generator": "Overpass API 0.7.62.1 084b4234",
  "osm3s": {
    "timestamp_osm_base": "2024-06-10T19:05:15Z",
    "copyright": "The data included in this document is from www.openstreetmap.org. The data is made available under ODbL."
  },
  "elements": [

{
  "type": "way",
  "id": 443668554,
  "nodes": [
    5385177806,
    11963746438,
    7007803230,
    511170655,
    7007803229,
    511166998
  ],
  "tags": {
    "natural": "coastline"
  }
},
{
  "type": "way",
  "id": 592262540,
  "nodes": [
    511167005,
    511166949,
    6468442690,
    7007803196,
    445269780
  ],
  "tags": {
    "natural": "coastline"
  }
```


Fig3. Result of the query from the previous figure (Fig2).

Explanation of the JSON Response

The result displayed in Figure 3 is a JSON response from the Overpass API query. This response includes detailed information about the coastline data points within the specified boundary. Below is a breakdown of the key components of the JSON response:

Metadata Section

- **"version": 0.6**: Specifies the version of the OSM data format.

- **"generator": "Overpass API 0.7.62.1 084b4234"**: Indicates the version of the Overpass API used to generate the response.

- **"osm3s"**: Contains metadata about the OSM data.

- **"timestamp_osm_base": "2024-06-10T19:05:15Z"**: The timestamp of the OSM data used for the query.

- **"copyright": "The data included in this document is from www.openstreetmap.org. The data is made available under ODbL."**: Provides copyright information and data licensing details.

Elements Section

The "elements" array contains multiple elements, each representing a part of the coastline data.

Individual Elements

- **"type": "way"**: Indicates that the element is a "way", which is a sequence of nodes forming a polyline.

- **"id": 443608554**: The unique identifier for the way.

- **"nodes"**: An array of node IDs that make up the way. Each node represents a specific point along the coastline.

For example: [538517786, 1196743648, 700870323, 511170655, 700870229, 511166998]

- **"tags"**: Contains key-value pairs describing the attributes of the way.

"natural": "coastline": Indicates that this way is tagged as a coastline.

Multiple Ways

The JSON response can contain multiple ways, each with its own set of nodes and tags. For example:

{

"type": "way",

"id": 443608554,

"nodes": [538517786, 1196743648, 700870323, 511170655, 700870229, 511166998],

"tags": {"natural": "coastline"}

}

{

"type": "way",

"id": 592262540,

"nodes": [511170705, 641166949, 641166975, 704870306, 704870316, 442569780],

"tags": {"natural": "coastline"}

}

Interpretation

The JSON response from the Overpass API provides a structured representation of the coastline data, allowing users to have a first representation of the coordinates and attributes of each segment of the coastline.

This figure illustrates the output of the Overpass API query, showing the JSON data structure containing the coastline details.

Then I used this API endpoint to create a code to generate a GeoJSON file with all data from the OSM coastline for the desired bounding box.

Example Python code used to generate GeoJSON file: Example Python code to generate GeoJSON file with OSM coastline data points

Based on this feature, I implemented an interactive map: [Ireland and more with OSM coastline visualized with red lines]

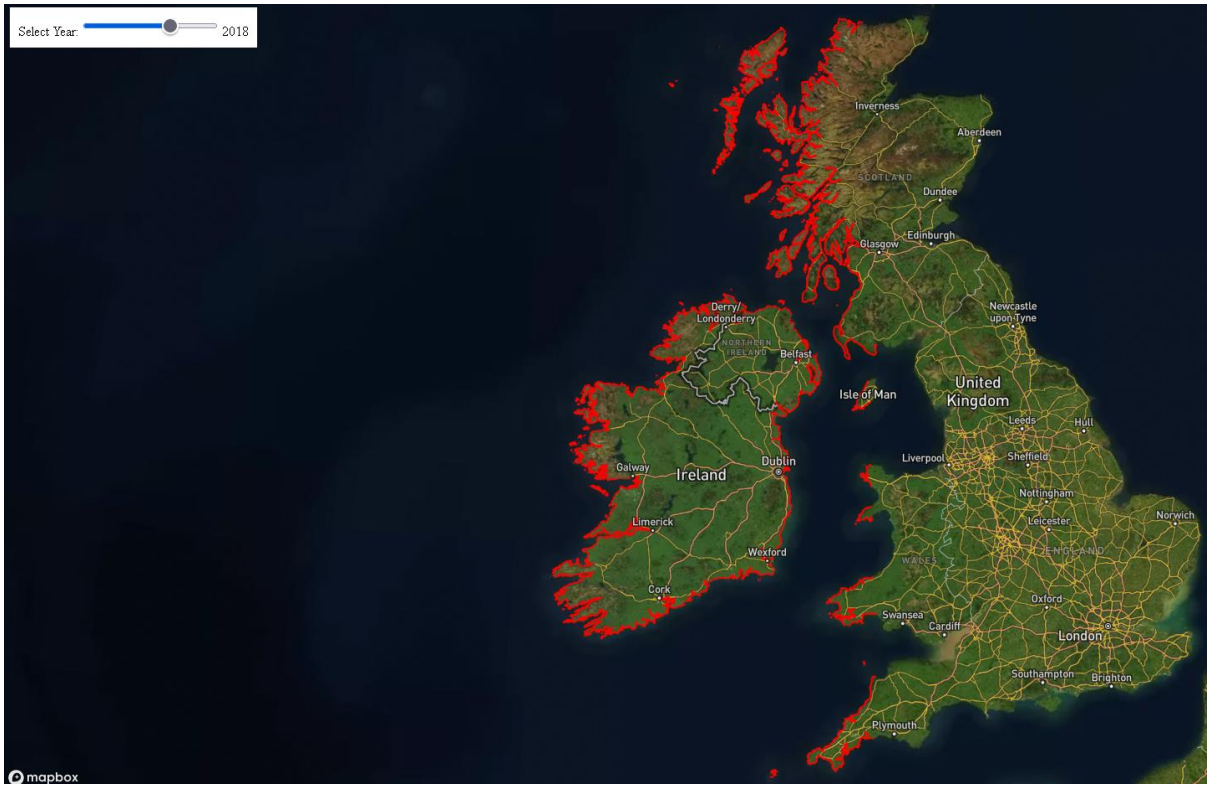

Fig4. Ireland and more with red lines representing the OSM coastline data points extracted from GeoJSON.

Key Features of the Interactive Map

- **Interactive Slider**: At the top left of the map, there is a slider allowing users to select different years. This feature provides a temporal aspect to the visualization, enabling the observation of coastline changes over time. However, it is important to note that this data corresponds to the JSON entries made by contributors each year, which may not always be precise.

- **Red Lines Representing Coastlines:** The coastlines are marked with red lines, clearly delineating the boundaries of land and water. These lines are derived from the OSM coastline data, providing an accurate representation of the coastal areas.

- **Geographical Coverage:** The map covers the entirety of Ireland and significant parts of the United Kingdom, including England, Scotland, and Wales.

- **Mapbox Integration:** The interactive map is built using Mapbox, a powerful tool for creating custom maps. Mapbox's capabilities enhance the user experience by allowing smooth zooming, panning, and interaction with the map elements.

This figure demonstrates the practical application of OSM coastline data in creating an interactive map, offering a first representation of the coastline but could not be accurate sometimes

This provides a first representation of the coastline, but it is not accurate:

Fig5. Example showing the inaccuracy of the OSM coastline data.

It's because contributors do their best to represent the coastline, but sometimes the datas are not accurate. Based on this conclusion, I searched for different research articles to solve my issues and found a useful article, which I will discuss in the next section.

2. SWED and Introduction to Shoreline Extraction

SWED article Sentinel-2 Water Edges Dataset (SWED)

SWED contains images captured by the European Space Agency's Sentinel-2 satellites between 2017 and 2021.

They used a convolutional neural network to transform original raw images into a segmentation map to clearly separate the water and the mainland.

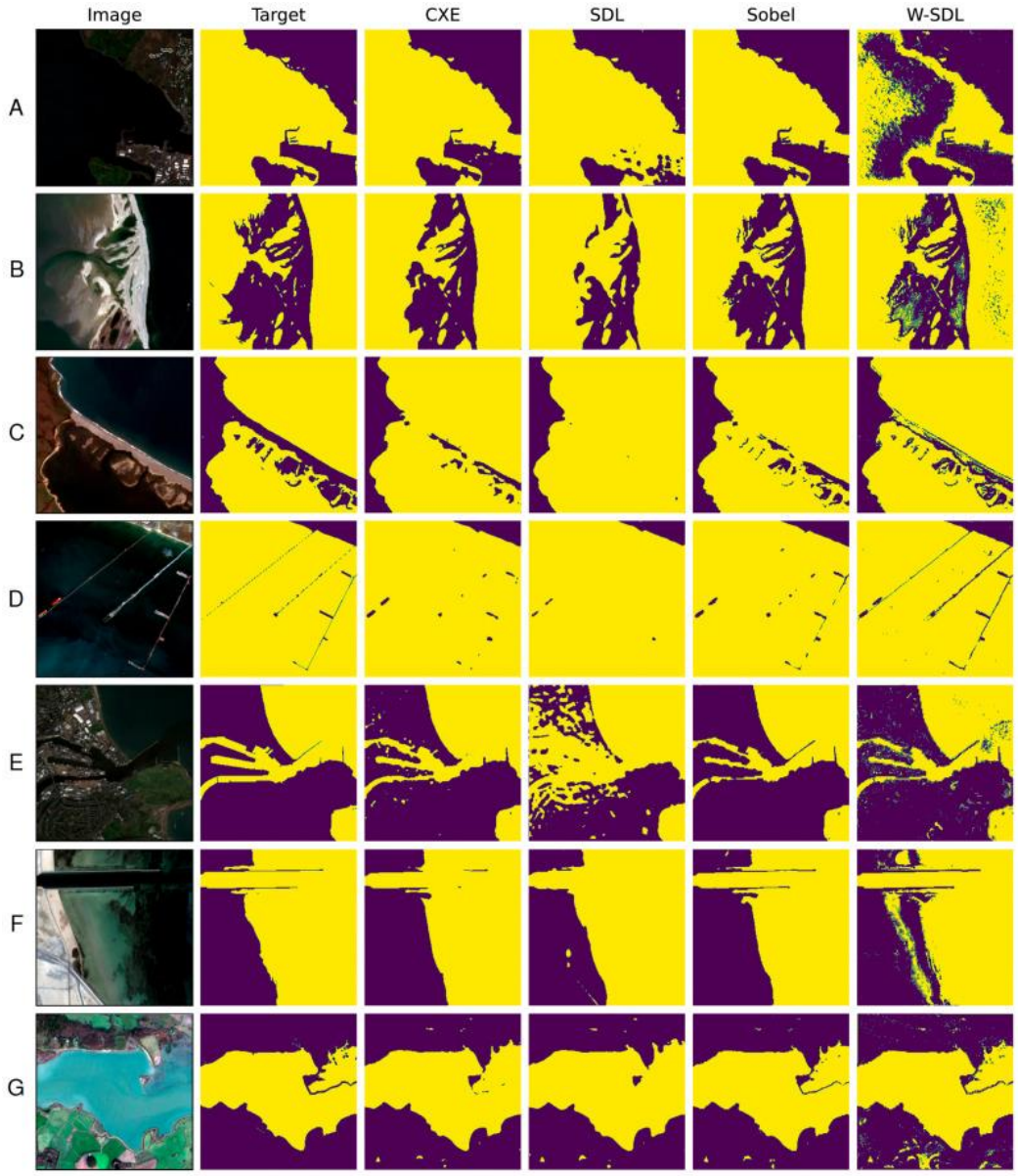

Fig6. [Extracted from SWED article, Fig6] Visualized segmentation results. The first and second columns depict the source image and the target label, respectively. The subsequent columns depict the predicted segmentation map from different models identified by the loss function used during training, e.g., CXE: Categorical Cross-entropy Loss, SDL: Sørensen–Dice Loss, Sobel: Sobel-Edge Loss, and W-SDL: Weighted-Sørensen–Dice Loss.

With this new dataset for training and benchmarking coastline extraction models, new articles have used the SWED dataset to explore different possibilities for shoreline extractions.

I will focus more on the following article: Automated Coastline Extraction using Edge Detection Algorithms

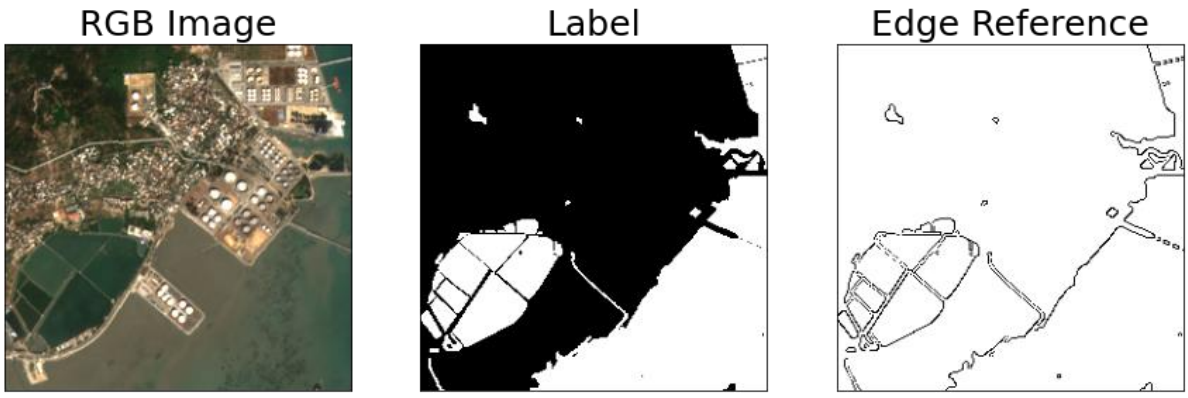

Fig7. [Extracted from SWED Edge detection article, Fig1] Example of a test image, binary land/water map, and coastline edge obtained from SWED.

As mentioned on the ARXIV website, the codes to reproduce the results of this paper can be found here: GITHUB SWED EDGE DETECTION

In the Jupyter notebook code, they analyzed the effectiveness of four edge detection algorithms to detect coastlines: canny_ed, scharr_ed, sobel_ed, and prewitt_ed, and concluded that the Canny edge detection method produced the best results.

The figure consists of several columns and rows, each serving a specific purpose in illustrating the edge detection results. The columns include the RGB Image, Label, and Edge Reference. The RGB Image represents the original satellite image in full color. The Label is a binary land/water map where land is shown in white and water in black, providing a clear distinction between the two. The Edge Reference column highlights the coastline edge, showing the boundary between land and water as detected by various algorithms. Each row in the figure represents a different test image from the dataset, offering multiple examples of how the algorithms perform.

Visually comparing the RGB Image and the Label helps to understand the accuracy of the segmentation process. The first two columns provide this comparison, allowing us to see how well the binary map represents the actual land and water areas. The Edge Reference column, on the other hand, provides a clear view of the extracted coastline edges, making it easy to assess the performance of the edge detection algorithms.

Among the edge detection algorithms tested, the Canny Edge Detection algorithm is noted for its superior performance. It effectively detects a wide range of edges in images, producing the best results among the algorithms tested. In contrast, the Scharr, Sobel, and Prewitt edge detection algorithms, while also useful, focus more on highlighting gradients and were less effective compared to the Canny algorithm.

Based on this, I tried to create a new prototype using Canny edge detection.

3. The New Prototype: Superimposing Images

Combine different techniques such as superimposing grayscale images with a popup system in an interactive map to compare the shoreline evolution in a visual representation.

I created a small prototype with a popup system in Dounam Bay: [Dounam Bay prototype]

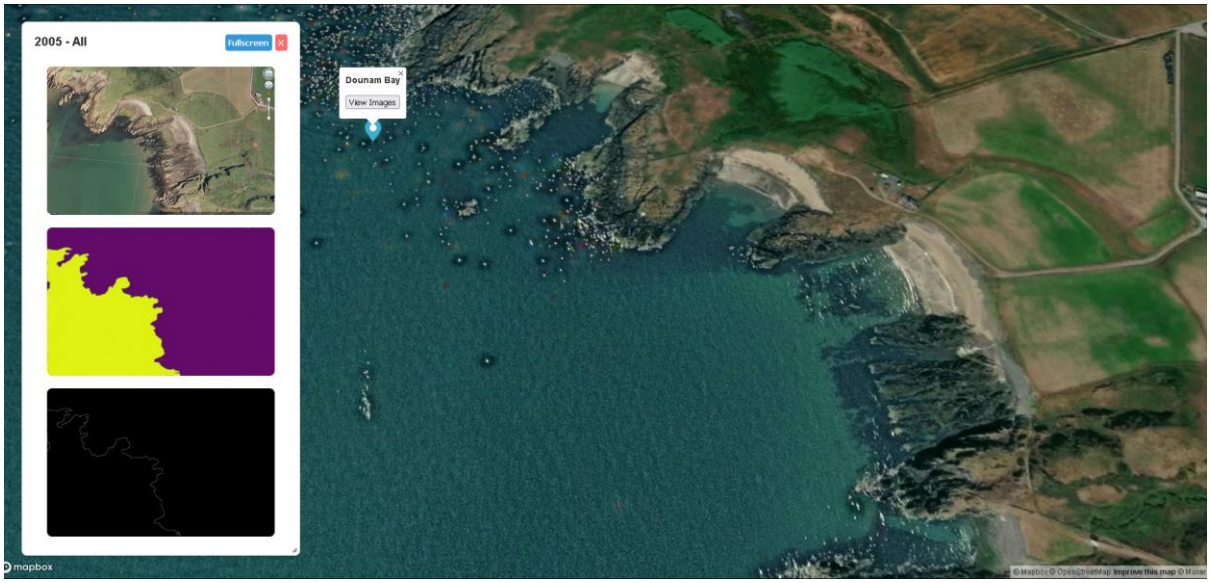

Fig8. Popup to compare from the latest map with multitemporal data.

Figure 8 illustrates a prototype developed to visualize shoreline changes over time using a combination of techniques such as superimposing grayscale images and integrating a popup system. This prototype is specifically implemented for Dounam Bay and provides a comparative visual representation of the shoreline evolution.

The prototype developed for visualizing shoreline changes over time employs a combination of techniques such as superimposing grayscale images and integrating a popup system in an interactive map. Specifically implemented for Dounam Bay, this prototype provides a comprehensive visual representation of shoreline evolution. The interactive map, developed using Mapbox, displays the geographic area of Dounam Bay, allowing users to navigate, zoom in and out, and explore different sections of the shoreline. The prototype features a popup system that enables users to view images from different years. When users click on the popup marker, it displays images and relevant data for that location, facilitating a comparison of shoreline changes over time. The images in the popup are grayscale representations of the shoreline at different times, which helps highlight changes, making it easier to identify areas of erosion and accretion. Additionally, there is a temporal slider at the top left of the map that allows users to select different years, adding a temporal dimension to the visualization and enabling users to observe how the coastline has evolved over the years.

Fig9. Example of superimposing grayscale images with a popup system in an interactive map to compare the shoreline evolution in a visual representation.

How did I generate the different grayscale images?

Firstly, I used a software named Gimp for manual labeling of the images. This process involves applying one color to represent land and another color to represent water.

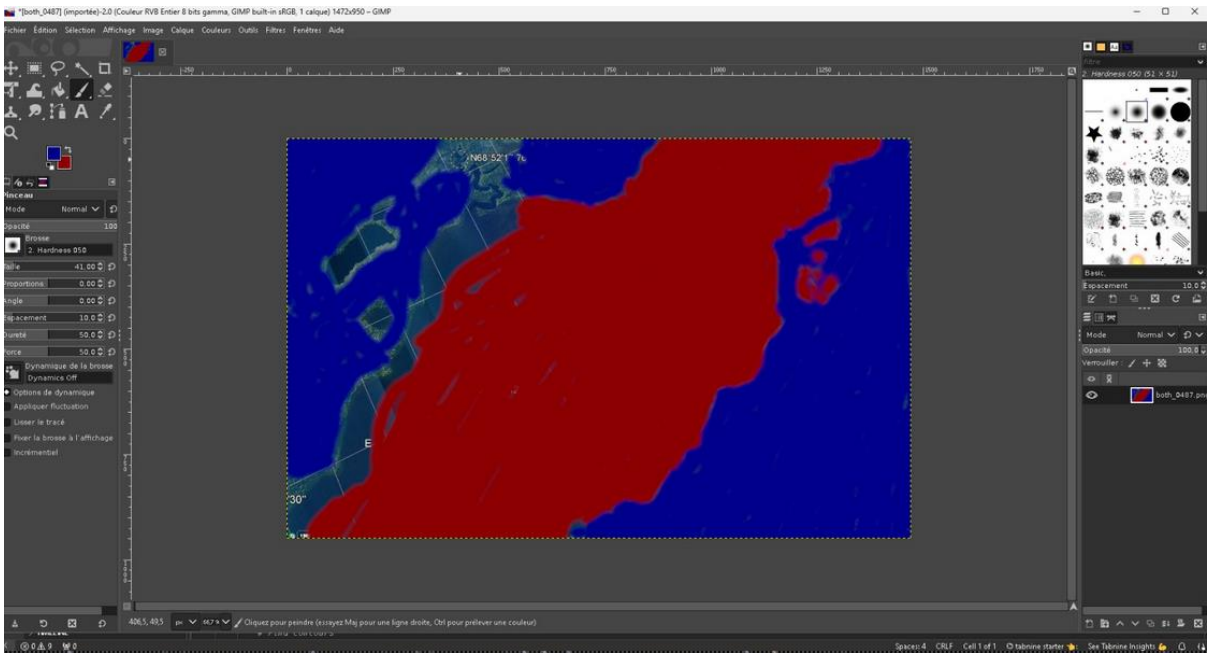

Fig10. Manual labeling of an image to identify the mainland and the water using Gimp.

Figure 10 shows how I used Gimp to manually color different areas, facilitating the distinction between land and water in satellite images. This method allows for accurate grayscale images necessary for analyzing shoreline changes.

```python
# Load the image
image = cv2.imread(image_path)

# Check if the image is loaded correctly
if image is None:
    print(f"Error loading image: {image_path}")
    continue

# Convert to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Apply threshold to create a binary image
_, binary = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)

# Detect edges using Canny
edges = cv2.Canny(binary, 100, 200)

# Find contours
contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# Create a black image
coastline_image = np.zeros_like(gray)

# Draw the contours in white
cv2.drawContours(coastline_image, contours, -1, (255, 255, 255), 1)
```

Fig11. Python code using Canny edges to generate grayscale images.

Next, I used Python with the OpenCV library to convert the manually labeled images into grayscale images. The following steps outline the process:

- **Loading the image:** The image is loaded using OpenCV's imread function.

- **Converting to grayscale:** The image is converted to grayscale using cvtColor.

- **Applying a threshold:** A binary threshold is applied to create a binary image, which separates the land and water.

- **Edge detection:** Canny edge detection is used to detect the edges in the binary image.

- **Finding contours:** The contours of the detected edges are found using findContours.

- **Creating a black image:** A blank image is created to draw the contours.

- **Drawing contours:** The contours are drawn on the black image in white.

This process is illustrated in the Python code snippet in Figure 11, where each step is implemented to convert the manually labeled images into grayscale images for further analysis.

Fig12. Dounambay Grayscale July 2011 generated with the Python code in the previous figure (Fig11).

Manual labeling of a picture with Gimp takes approximately 8 minutes per picture, so I tried to create an auto-labeling feature with machine learning.

Firstly, I trained a classification algorithm, detailed in the next section.

4. The Auto-labeling Feature for Google Earth Screenshots

To streamline the labeling process, I captured approximately 1200 different screenshots from Google Earth Studio. These images served as the dataset for training a machine learning model to automatically label the shoreline images.

The Github repositository is here : Github repo

The zip archive which conatins the classification folder is here : Zip archive

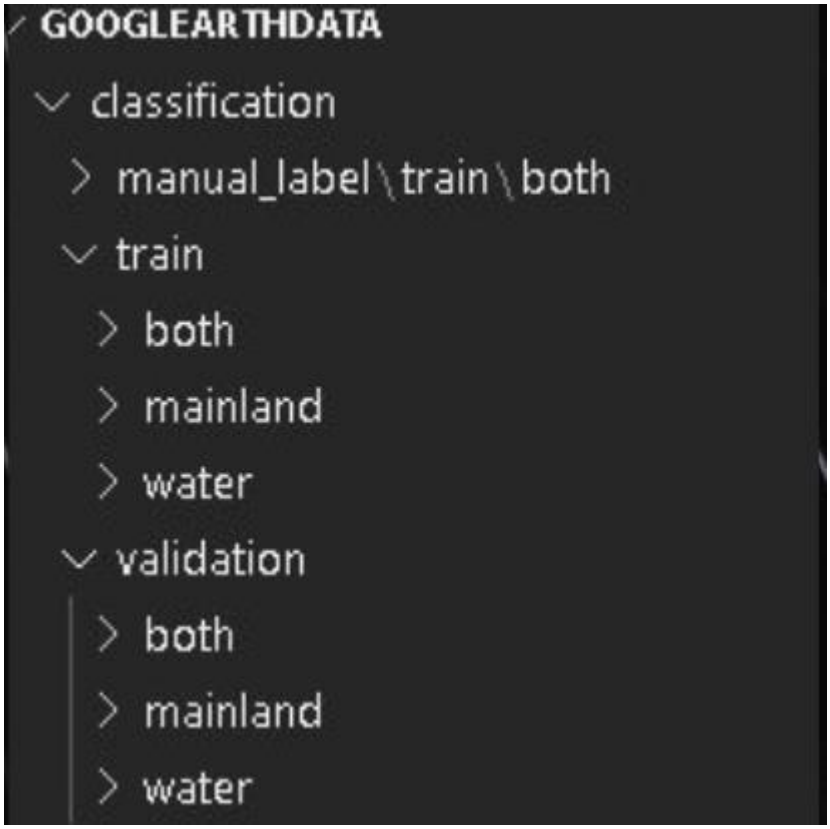

Fig13. Directory structure of the Google Earth data used for training and validation.

Figure 13 shows the directory structure of the dataset. The data is organized into three main categories: both (for images containing both land and water), mainland (for images containing only land), and water (for images containing only water). Each category is further divided into training and validation sets, ensuring a structured approach to model training and evaluation.

This organization facilitates the training of a classification algorithm that can automatically label the images, significantly reducing the time and effort required for manual labeling.

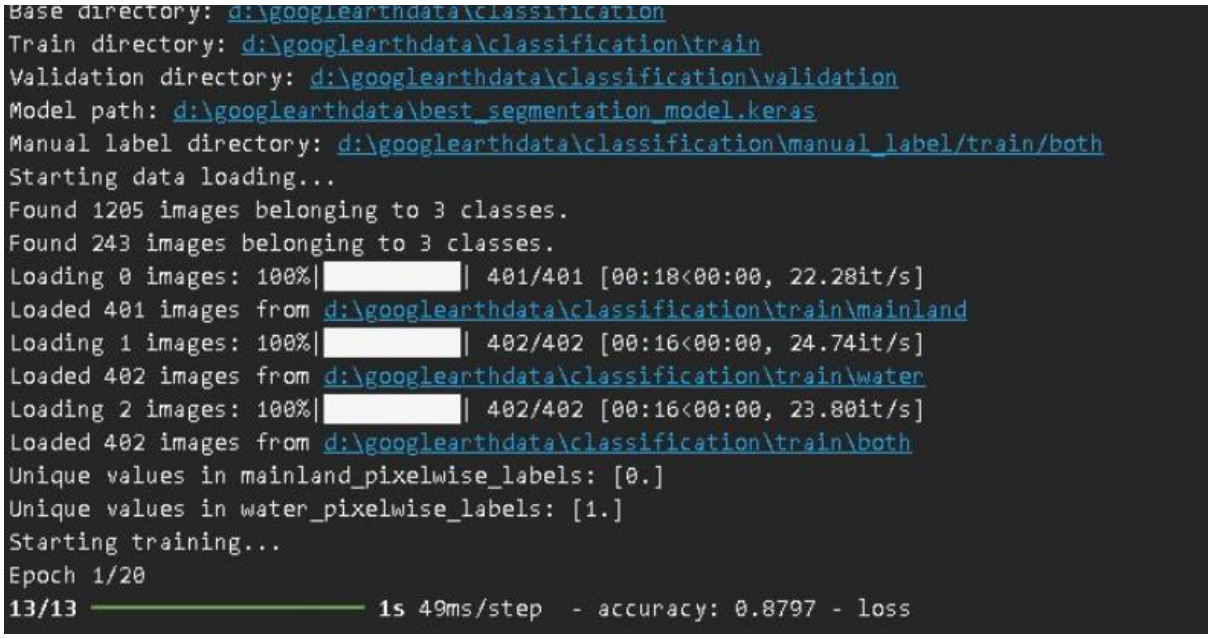

Fig14. Screenshot showing the process of loading data for training the classification model. The data is organized into training and validation directories with three classes: mainland, water, and both. The model begins training with the loaded data.

Figure 14 illustrates the process of loading the training and validation data. The dataset contains 1205 images divided into three classes: mainland, water, and both. The screenshot shows the loading process for each class and the start of the training process, with the model achieving an accuracy of 87.97% in the initial epochs. This structured approach to data management and model training ensures the robustness and reliability of the auto-labeling feature.

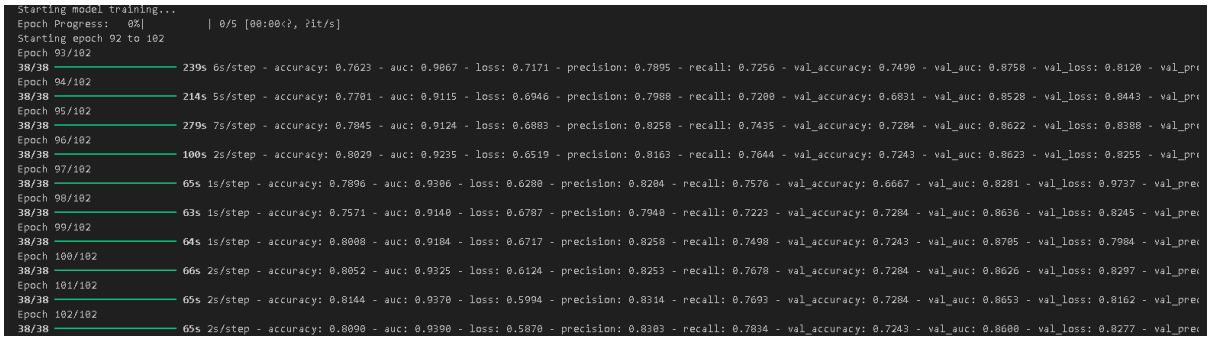

Fig15. Training the classification model over 102 epochs. The screenshot shows various performance metrics, including accuracy, precision, recall, and loss for both training and validation datasets. The model's accuracy improves steadily, indicating effective learning.

Figure 15 shows the training process over 102 epochs, highlighting the model's performance metrics such as accuracy, precision, recall, and loss. The model demonstrates steady

improvement in accuracy and other metrics, indicating effective learning and the potential for reliable auto-labeling of shoreline images.

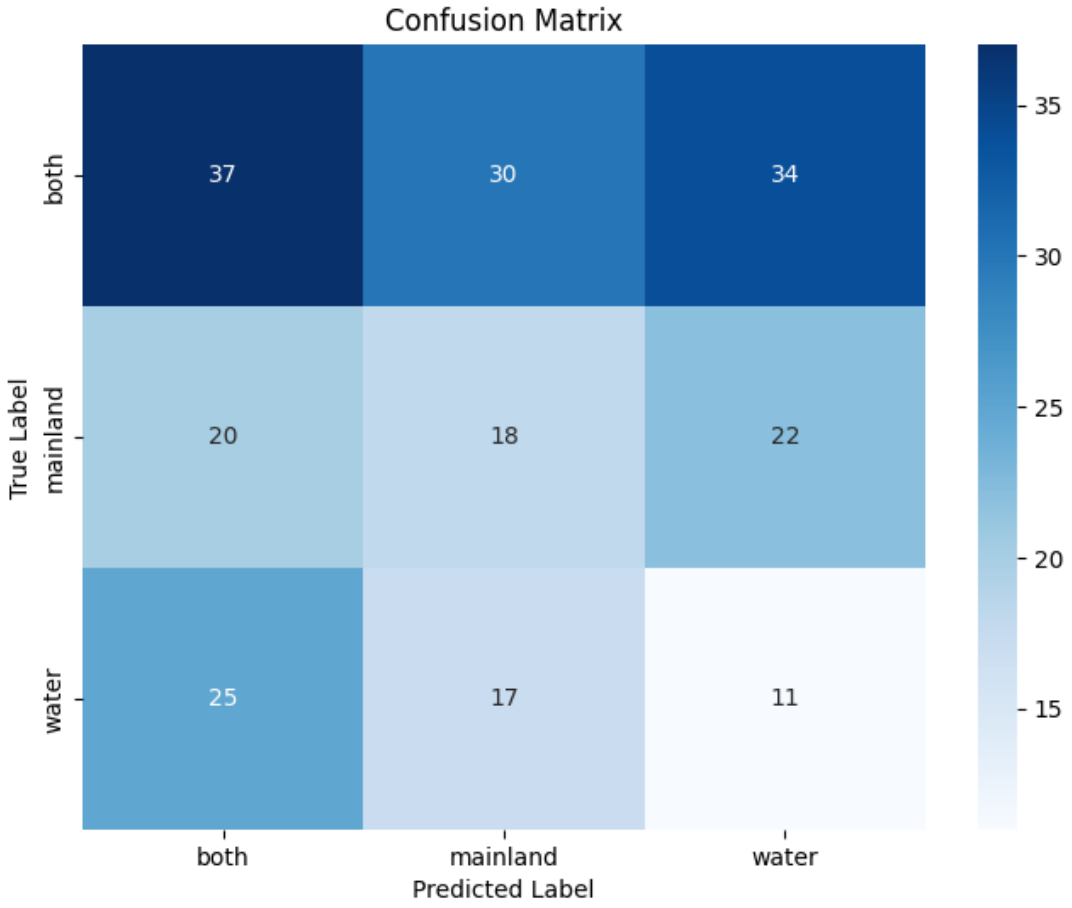

Fig16. Confusion matrix illustrating the performance of the classification model. The matrix shows the true labels versus the predicted labels for the three classes: both, mainland, and water. This provides insight into the model's accuracy and areas for improvement.

Figure 16 provides a detailed look at the confusion matrix for the classification model. The matrix compares the true labels against the predicted labels for the three classes: both, mainland, and water. It highlights the model's strengths and weaknesses, showing where misclassifications occur. For instance, the model tends to confuse images of water and both, indicating areas where further refinement is needed.

To better understand why the loss is high, the following figure illustrates the model's predictions:

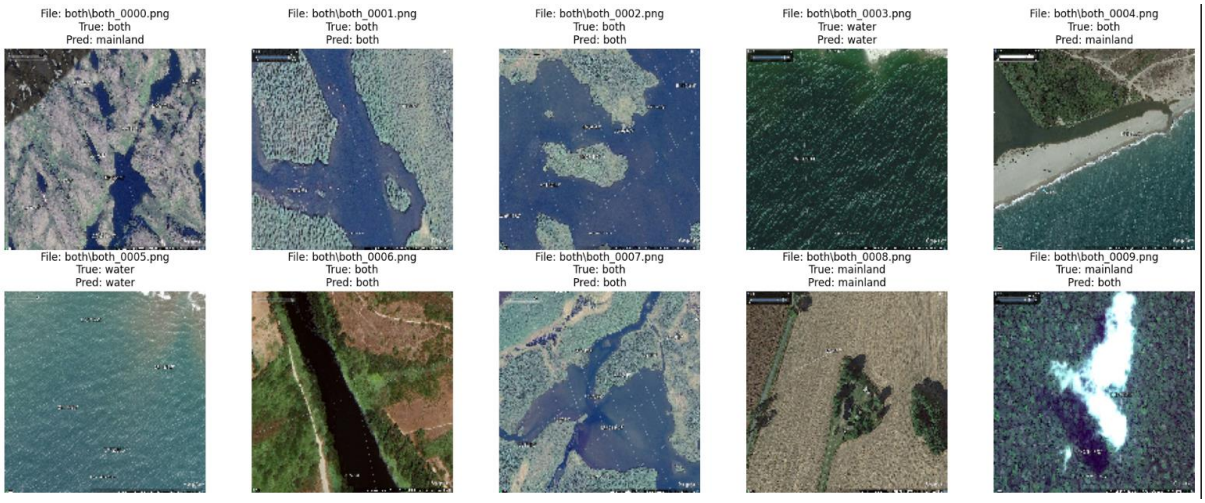

Fig17. Examples of predictions from the classification model. Each image shows the true label and the predicted label, demonstrating where the model performed correctly and where it misclassified.

Figure 17 presents examples of the classification model's predictions. Each image displays the true label and the predicted label, highlighting instances where the model correctly identified the category and where it made errors. This visual representation helps in understanding the model's performance and identifying areas for improvement.

You can find two different repositories of code on GitHub: First version of my code, the second version: Second version of my code, and Version for Google Earth data.

Based on these results, I concluded that the classification method might not be sufficient, so I tried another machine learning algorithm: Unet.

5. Segmentation Image Processing with Weighted Cross Entropy

In this section, we explore segmentation image processing using a weighted cross-entropy loss function. This method helps to improve the accuracy of segmenting images into different classes, such as land and water.

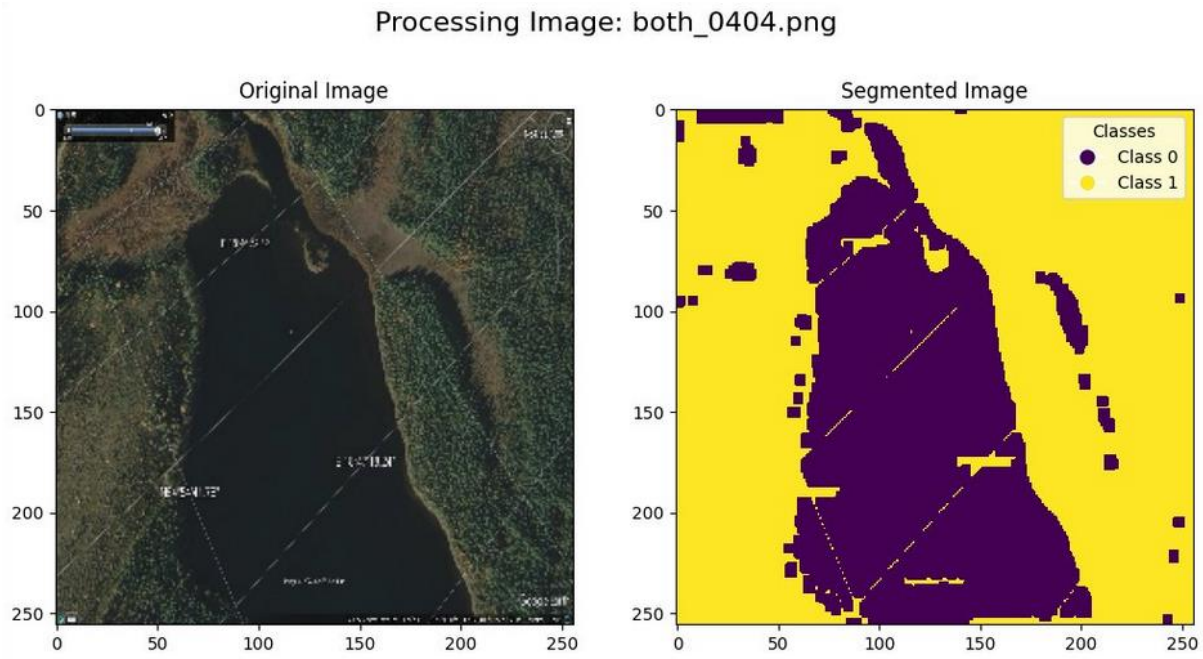

Fig18. Example of processing an image from the Google Earth Engine dataset. The left image is the original satellite image, and the right image is the segmented version, where Class 0 (land) is shown in purple and Class 1 (water) is shown in yellow.

Figure 18 illustrates the process of segmenting a satellite image into land and water classes using a machine learning model trained with a weighted cross-entropy loss function. The original image (left) is processed to produce a segmented image (right), which classifies each pixel as either land (Class 0) or water (Class 1).

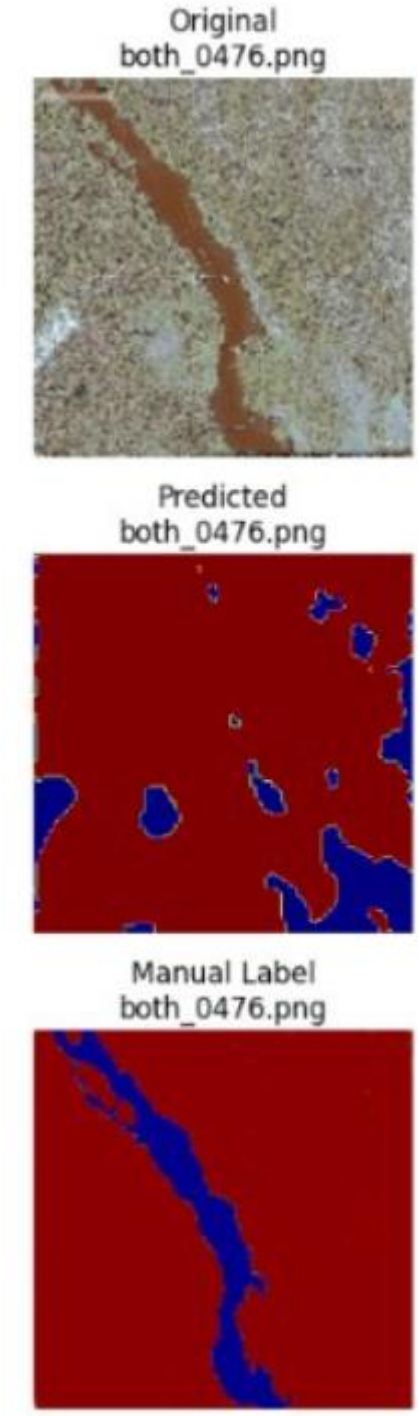

Fig19. Comparison of Original, Predicted, and Manual Label images for a test case. The top image is the original satellite image, the middle image shows the predicted segmentation by the model, and the bottom image is the manual labeling for reference.

To improve the model's accuracy, I implemented a manual loss function similar to the weighted cross-entropy loss. This approach allows the model to learn from manual labels whenever the loss on a picture is high, helping to understand the reasons behind wrong predictions and to reduce the overall loss.

However, I encountered several challenges. One major issue was the presence of a grid in the screenshots taken from Google Earth Studio Pro, which made it difficult for the UNet algorithm to accurately distinguish between water and mainland.

Additionally, not correlating the shorelines with data on tides and sea levels can lead to incorrect conclusions about the shoreline's evolution. It is crucial to integrate these datasets to ensure a comprehensive understanding of shoreline dynamics.

After discussing these issues with Hachem, he recommended using CoastSat, a package widely used by coastal engineers for shoreline detection and analysis.

6. CoastSat

The research article is available here: CoastSat research article

The GitHub repository code is available here: GitHub repository code

CoastSat also has a website: CoastSat website

The CoastSat website provides an interactive map to visualize coastal erosion in different locations.

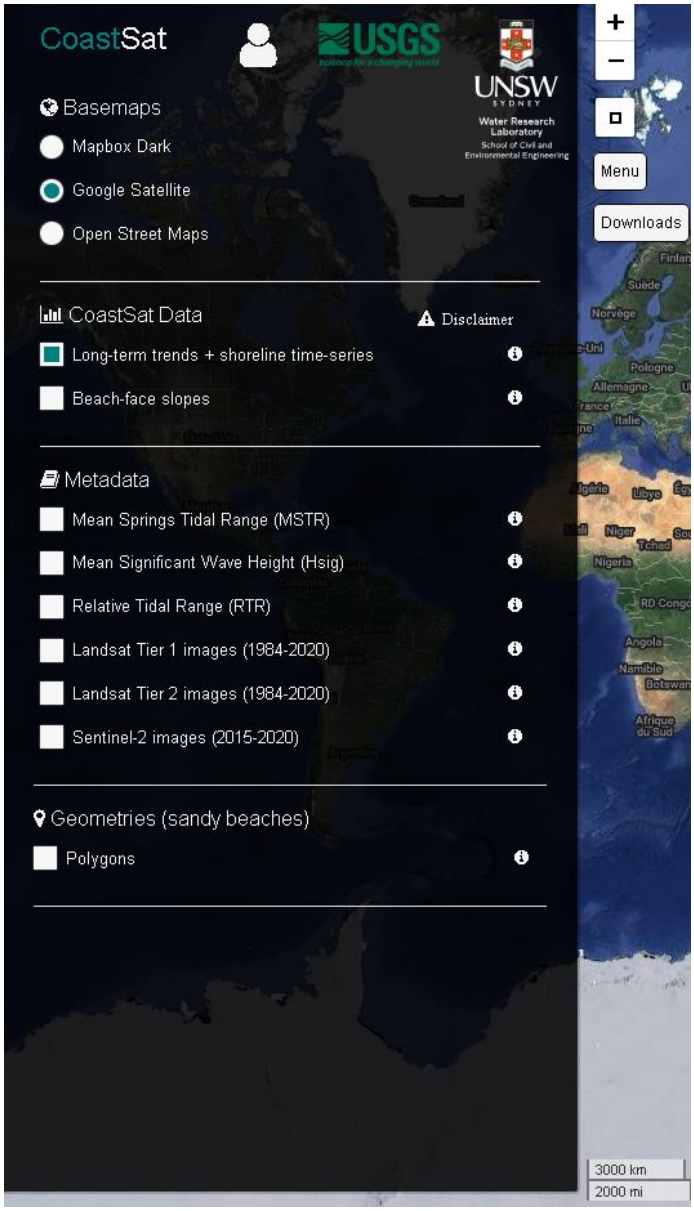

Fig20. Sidebar menu of CoastSat showing options for basemaps, CoastSat data, metadata, and geometries.

On the left side of the interface, we can see the sidebar menu of CoastSat where different parameters can be selected, such as basemaps, CoastSat data, and metadata. This menu also includes options for visualizing long-term trends, shoreline time-series, and beach-face slopes.

Additionally, the sidebar provides access to metadata like Mean Springs Tidal Range (MSTR), Mean Significant Wave Height (Hs), Relative Tidal Range (RTR), and satellite images from Landsat and Sentinel-2. A download button is available for data download specifications.

Integrating these datasets allows users to study and visualize coastal changes comprehensively, making it an invaluable tool for researchers and policymakers in coastal management and erosion mitigation.

Legends are also provided on the map, such as the trend (meter/year).

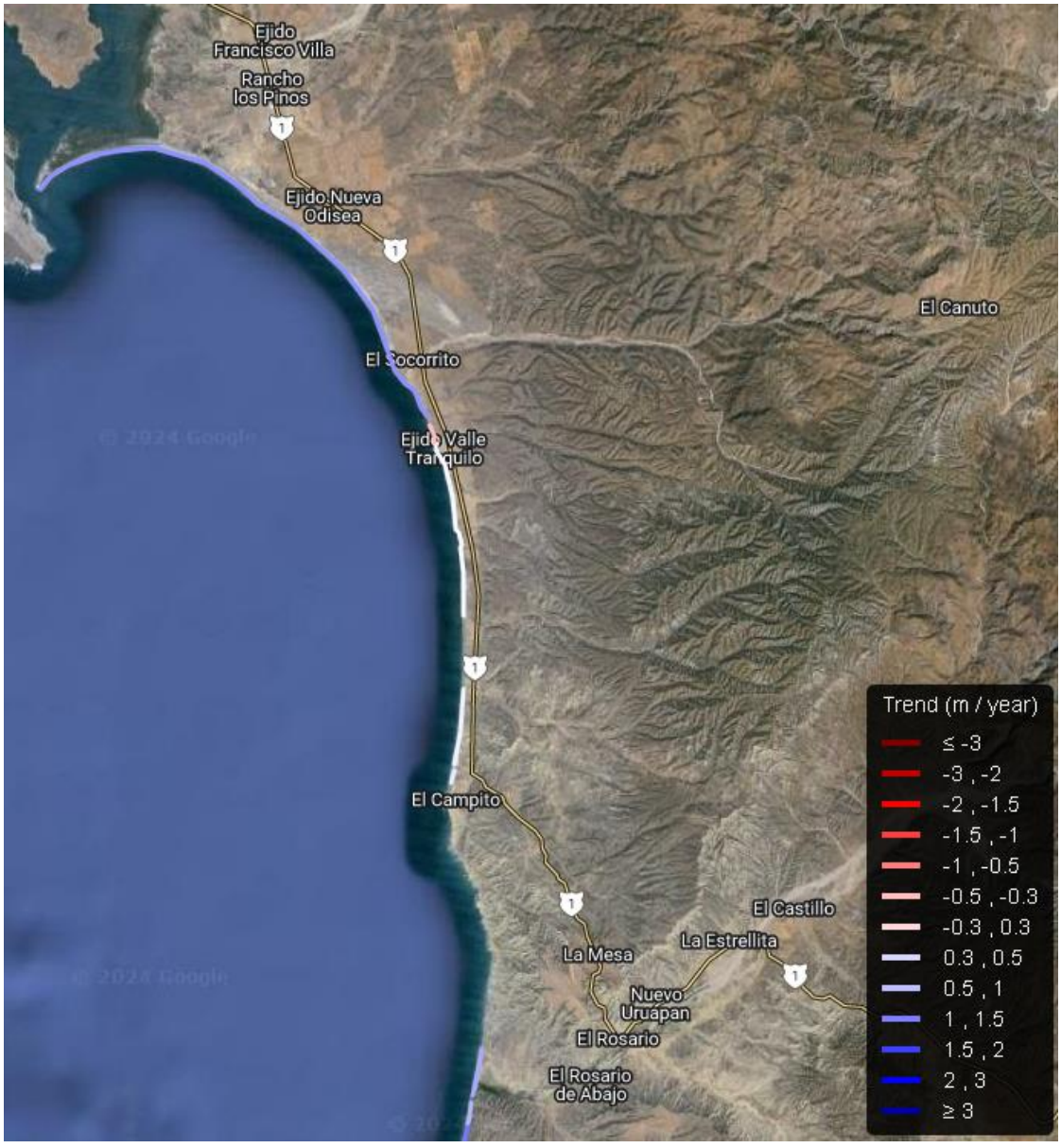

Fig21. Trend (m/year) along the coast represented with different colored lines.

As seen in Fig21, there are different colored lines along the coast. These colors represent various rates of coastal change, with the legend indicating the trend in meters per year. This allows users to quickly identify areas experiencing significant erosion or accretion.

These lines represent the shoreline of a specific area. If we click on a shoreline:

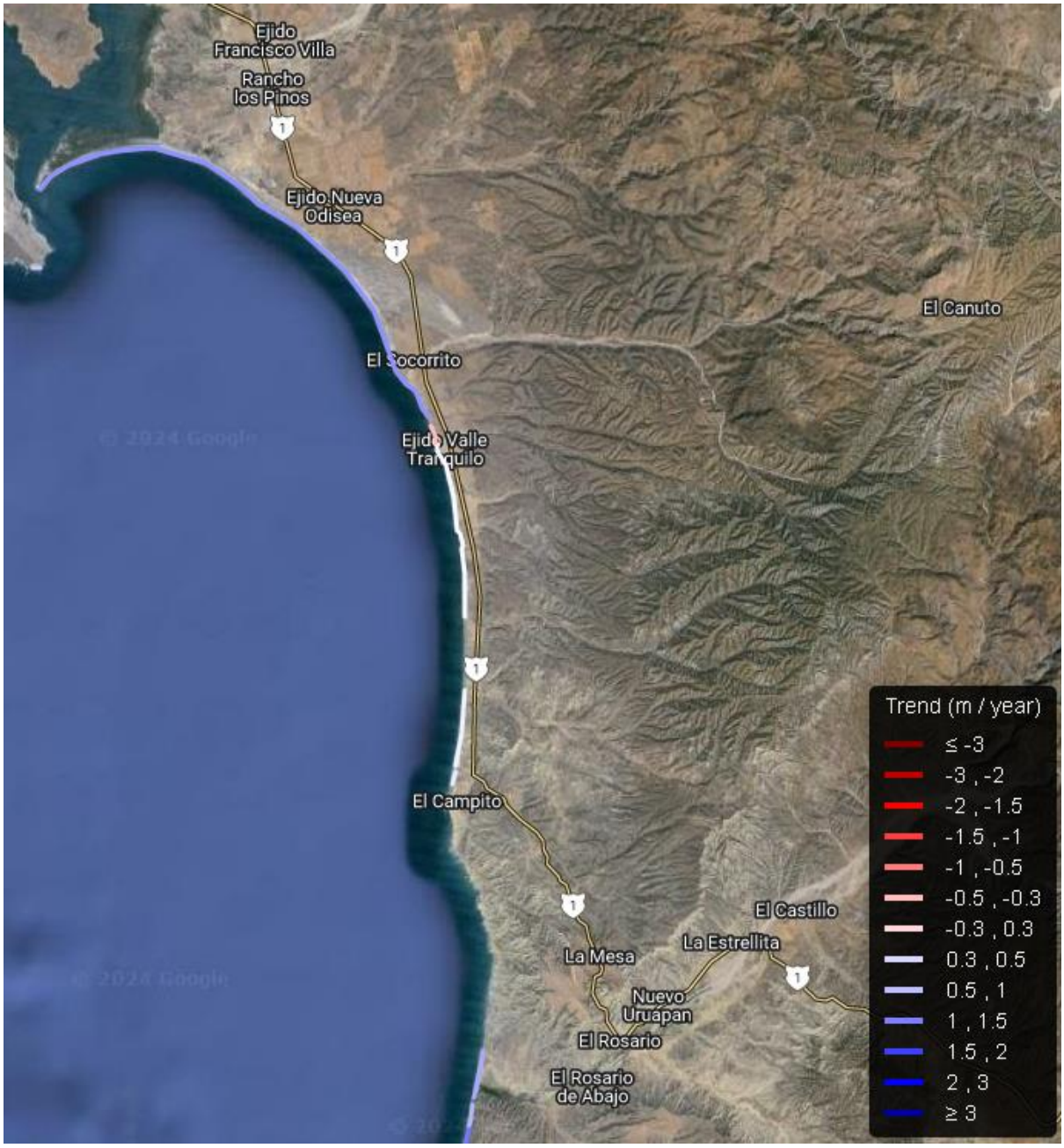

Fig22. Shoreline popup displaying mean trend and additional data.

As shown in Fig22, a popup appears when a shoreline is clicked. This popup provides detailed information about the selected segment, such as the mean trend (in meters per year) and the tangent of the beach slope (tanβ). Additionally, a button labeled "show transects" allows users to view transect data for more detailed analysis.

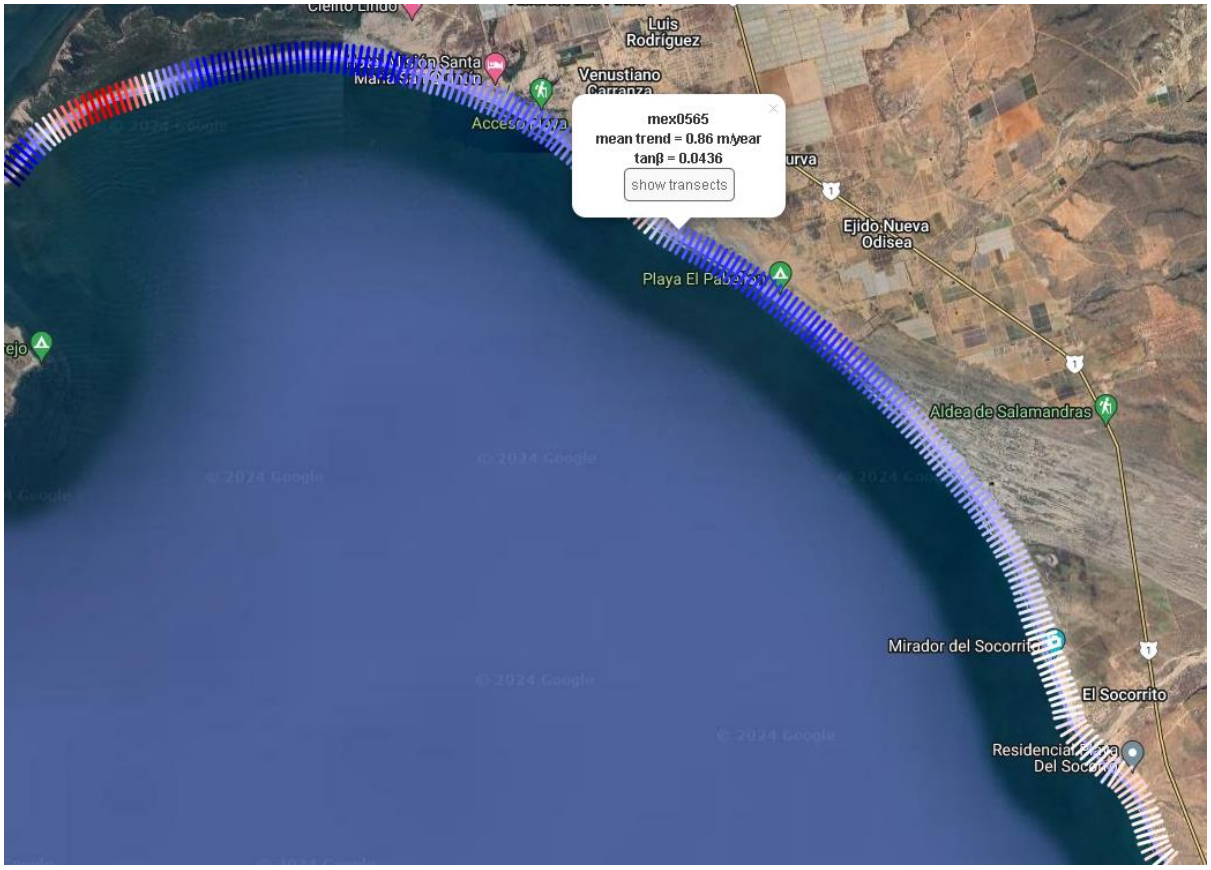

Fig23. Show transect clicked.

When the "show transects" button is clicked, all transects of the shorelines are displayed with a specific color related to the legend (trend in meter/year). This feature provides a visual representation of the shoreline's evolution, helping to understand the patterns of erosion and accretion over time.

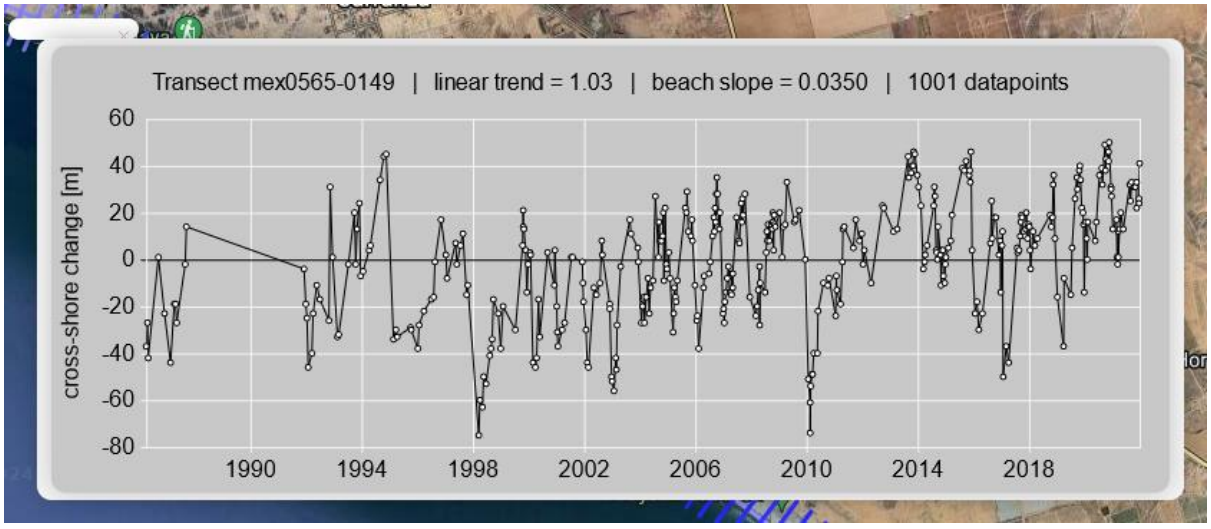

Fig24. Transect clicked.

When a transect is clicked, a popup is displayed with a cross-shore change chart to see the evolution over the years. As seen in Fig24, the chart provides a visual representation of changes in the cross-shore distance over time, indicating periods of erosion and accretion.

With CoastSat software, we can study specific areas. The software contains functions to retrieve images from different satellite missions and collections over 30 years, extract shorelines using a sub-pixel resolution technique, and generate shoreline animations to see the erosion over time. Transects are used in both main projects, subdividing a specific area to identify a general trend in that area.

They created a hazard summary focused on Narrabeen sandy beaches: Australian Beach Erosion and Coastal Flooding EWS focus on Narrabeen

7. My Prototype

Based on all previous work, I created a new prototype to address my research topic: Interactive Coastal Erosion Map with AI Predictions.

Note that my prototype is highly inspired by CoastSat and uses some CoastSat data (CoastSat website).

You can find the latest version of my prototype here: Prototype

Fig25. Earth Project Screenshot.

In this prototype, I used the Mapbox library to create an interactive map and implemented data from CoastSat. This map allows users to explore coastal erosion data and AI predictions

in an intuitive and interactive way, enhancing the understanding of shoreline changes over time.

After using CoastSat software, I started to make new data points on France.

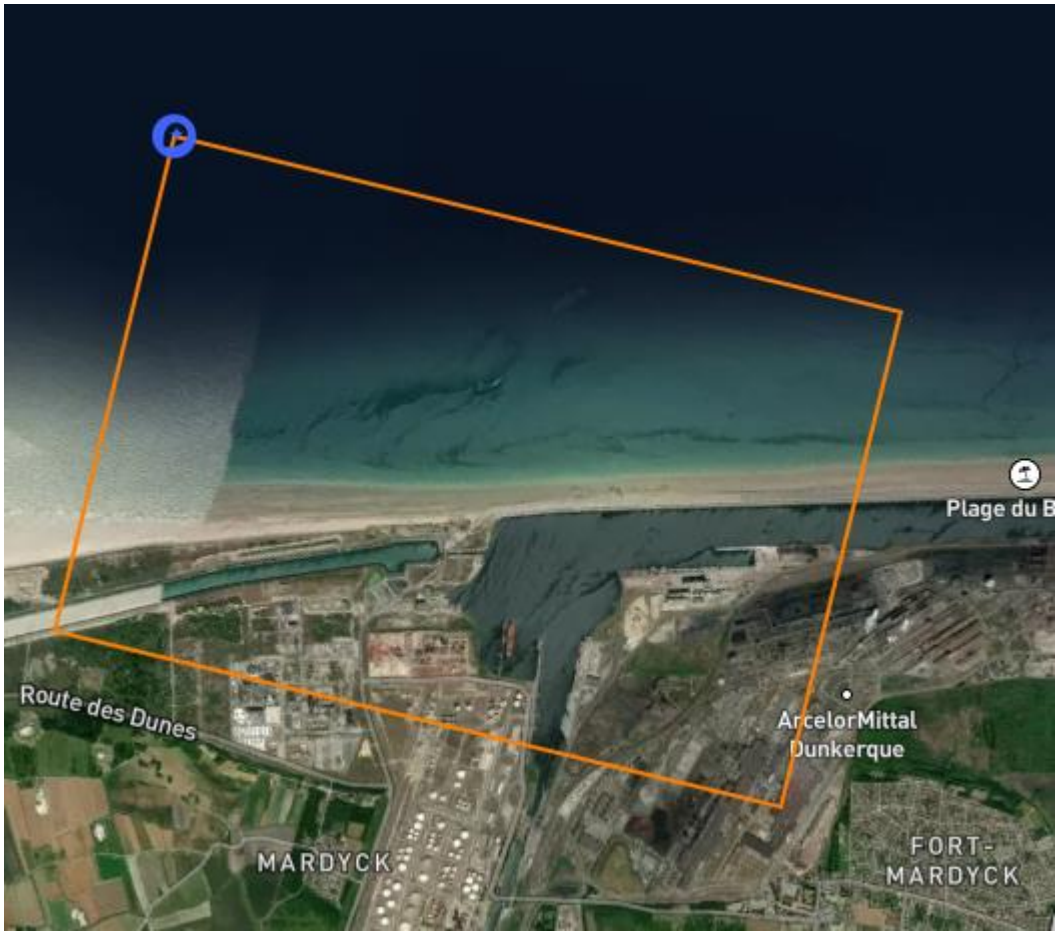

Fig26. A polygon study area in France.

This image shows a polygon study area on the coast of France. The polygon outlines the specific region where new data points are being collected and analyzed to study coastal erosion and shoreline changes using the methodologies and tools discussed earlier.

If you click on the custom blue marker:

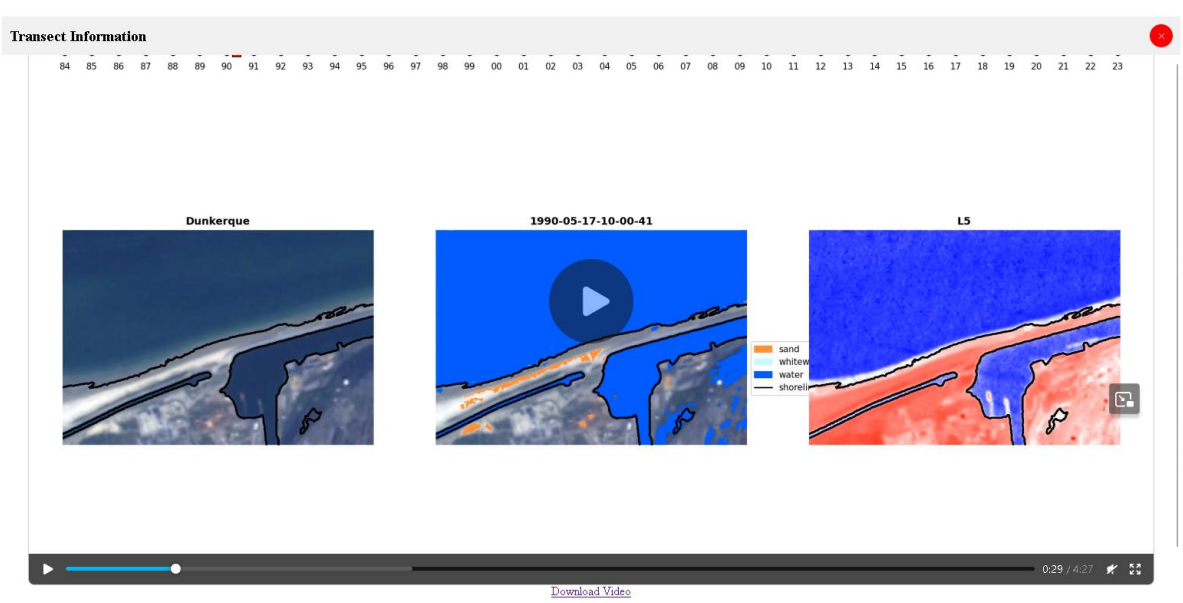

Fig27. Shoreline animation generated by CoastSat software.

With the shoreline animation, we can study a specific area with all satellite images to check the shoreline evolution over time. This animation helps visualize how the shoreline has changed and allows for a more detailed analysis of coastal erosion patterns.

At the bottom of the shoreline animation, there is a select button.

Fig28. Select Images to Compare.

This feature allows users to select two different satellite images for comparison. By using this tool, users can visually analyze changes in the shoreline between the selected dates, providing a clear understanding of the coastal erosion or accretion over time.

When an image is selected from the comparison tool, the interface shows detailed visualizations of the selected image.

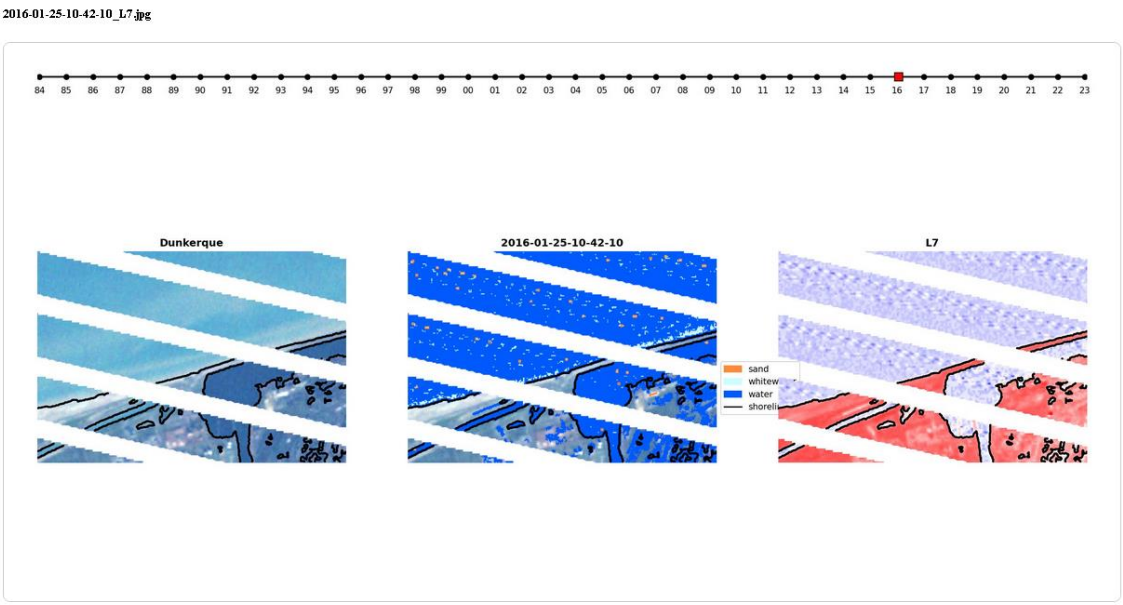

Fig29. Detailed visualization of the selected image from the comparison tool.

The selected image is presented with various layers such as sand, whitewater, water, and shoreline, allowing users to thoroughly investigate the coastal changes. This visual representation helps in identifying specific areas affected by erosion or accretion over the years.

Additionally, the sea level information is provided for the selected image to offer a more comprehensive analysis.

Sea Level from file1: 3269 mm

Sea Level from file2: 3271 mm

URL: https://www.sonel.org/spip.php?page=maregraphe&idStation=1758

Fig30. Sea level information of the first image.

This information includes the sea level readings at the time the image was captured, which is crucial for understanding the context of the shoreline changes. Users can also access the URL provided for more detailed sea level data from the SONEL website.

The comparison tool also allows users to view another selected image in detail, as shown below:

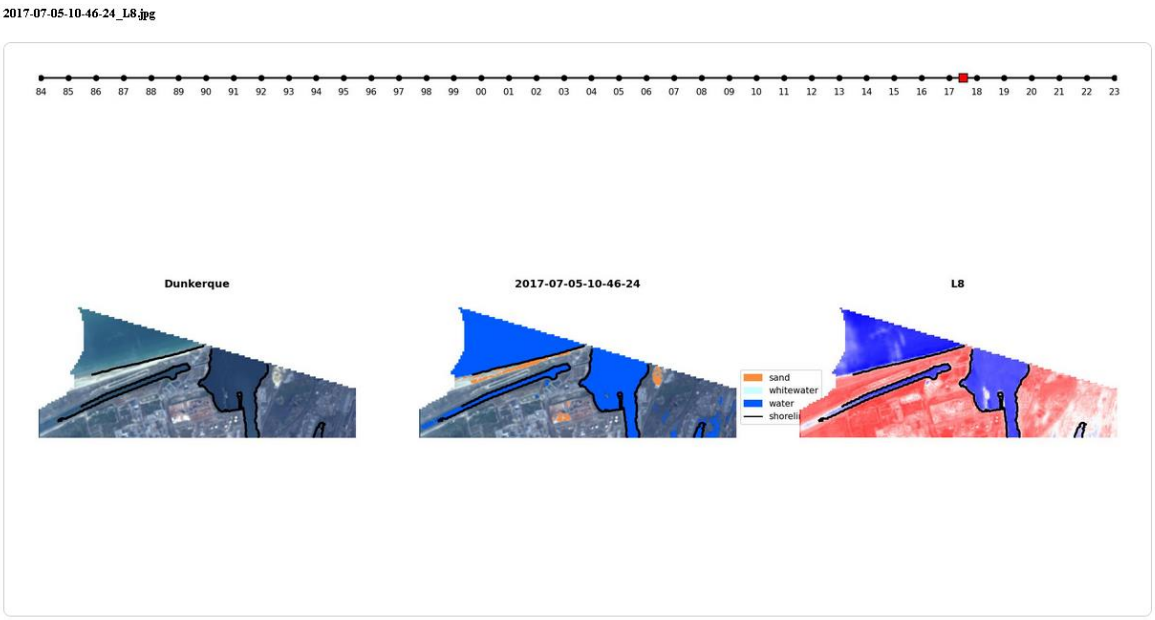

Fig31. Second image.

This second image provides additional insights into the coastal changes over time, with similar visual layers and information to facilitate a comprehensive comparison. By analyzing multiple images from different dates, users can better understand the dynamics of coastal erosion and accretion.

Finally, the sea level information is also provided for the second image:

Fig32. Sea level info of the second image.

The red button "Close Comparing Two Images Section" closes the comparing section, and the user returns to the section with the video and the select button if they want to compare other images.

I also implemented CoastSat data by downloading them to integrate them into my latest prototype.

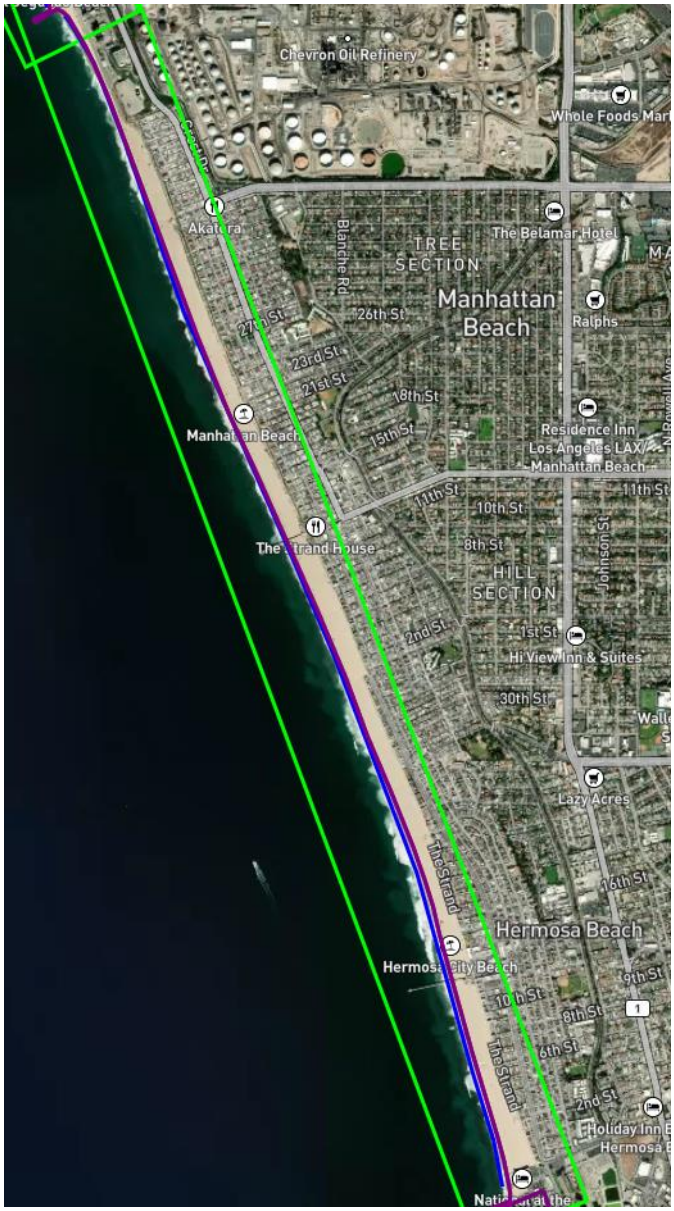

Fig33. Polygon new prototype.

In this figure, you can see the polygon representing the study area in my new prototype. The integration of CoastSat data allows for precise and detailed analysis of the shoreline, making it possible to visualize and monitor coastal changes with high accuracy.

Fig34. Purple line.

The purple line represents the OpenStreetMap (OSM) coastline extracted with the API. This integration allows for the visualization of current coastline data, making it easier to compare with other data sources and analyze coastal changes over time.

Fig35. Blue line.

The blue line represents the shoreline of the specific area inside the polygon.

When the shoreline is clicked, all transects of the shorelines are displayed (in red lines).

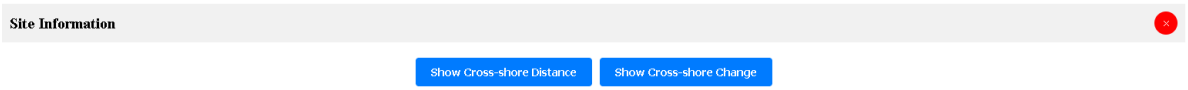

Fig36. Transect clicked on the new prototype.

This prototype features a "Site Information" section, where users can see specific details about the selected site. Additionally, two buttons labeled "Show Cross-shore Distance" and "Show Cross-shore Change" provide options for further data analysis. When clicked, these buttons display transects (red lines) that help in understanding the changes in the shoreline by

showing distance and change metrics across different years. This functionality enhances the analysis of coastal erosion by giving a clear visual representation of the changes over time.

Two buttons are displayed: "Show Cross-shore Distance" and "Show Cross-shore Change".

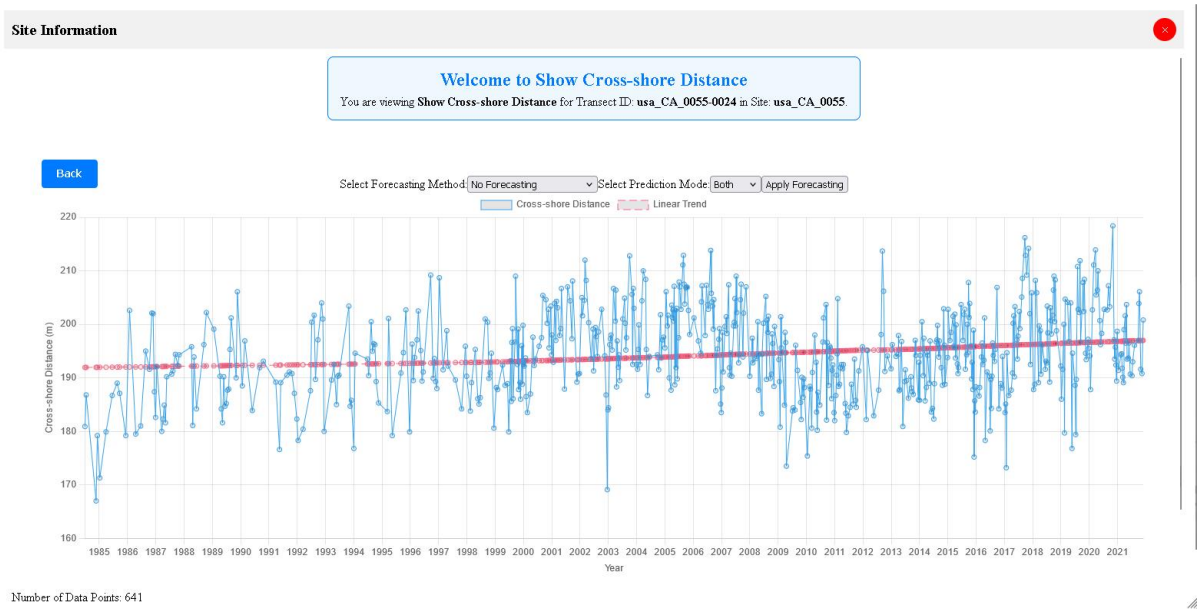

Fig37. Show cross-shore distance.

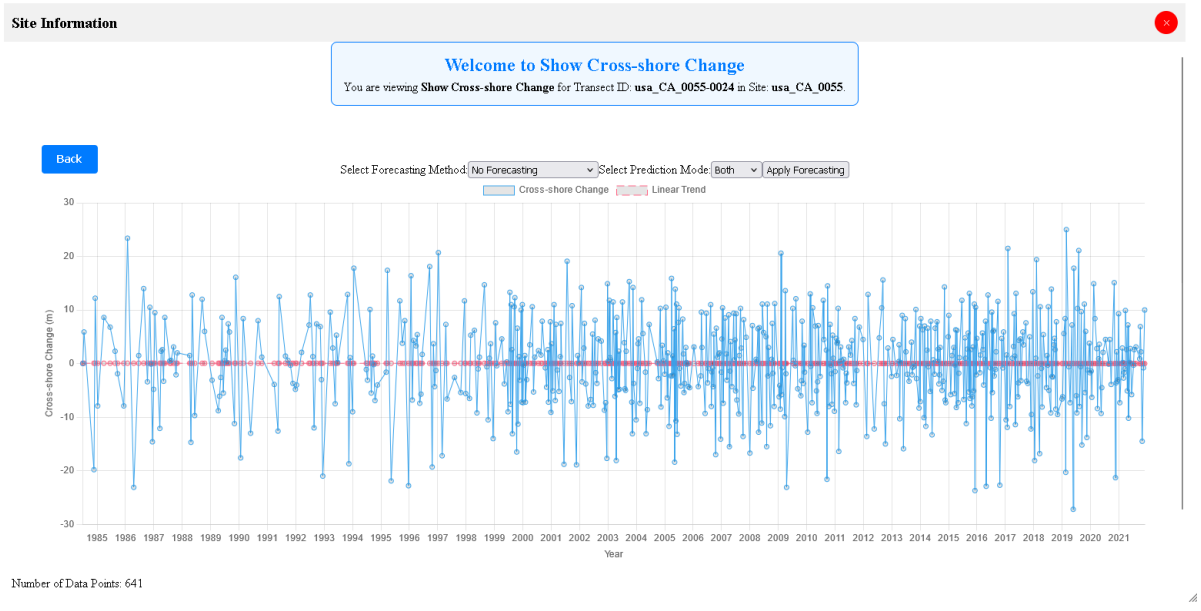

Fig38. Show cross-shore change.

These graphs provide detailed insights into the cross-shore distance and change over time. Additionally, there are two more buttons: "Select Forecasting Method" and "Select Prediction Mode", which are used for the time forecasting part detailed in the next section.

8. AI Predictions (Times Forecasting)

To implement this part, I created a webpage for time series analysis and prediction: AI predictions

The goal of this part is straightforward: Using existing data, I aimed to find a method to predict the past and future of the shoreline.

On this webpage, I implemented various forecasting methods such as Naïve, SNaïve, Seasonal Decomposition, State Space Model, and Holt-Winters directly on the time-series data collected from the CoastSat website

After implementing these methods, I applied them to the transect data to generate forecasts.

The Naïve method is a simple time series forecasting approach that assumes future values will be equal to the last observed values in previous cycles.

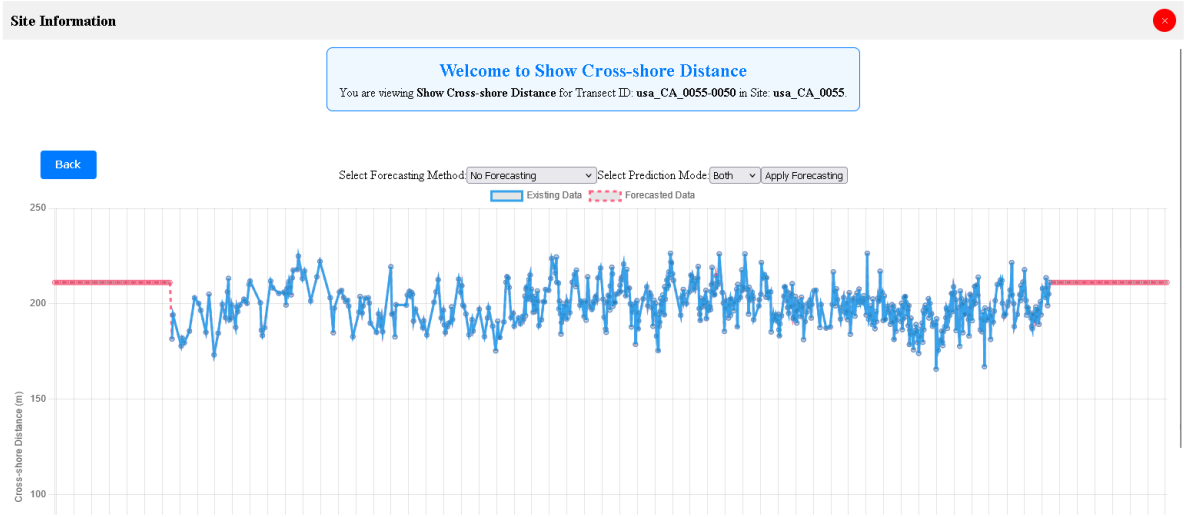

Fig39. Naive time forecasting.

The SNaive (Seasonal Naive) method predicts future values by assuming they will be equal to the last observed values from the same season in previous cycles.

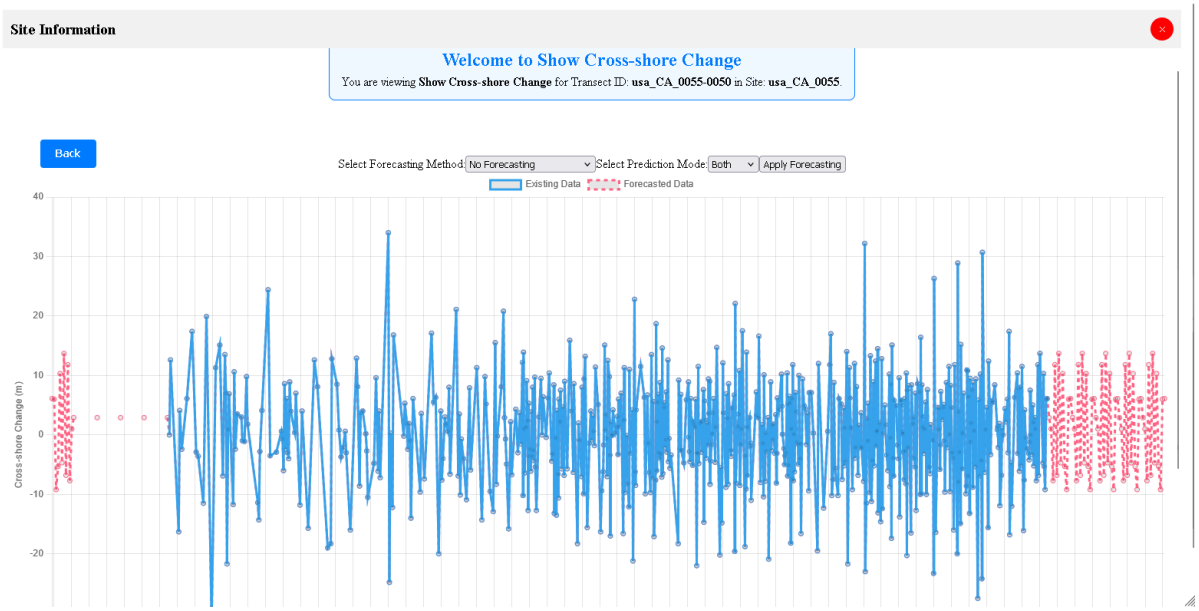

Fig40. Seasonal Naive time forecasting.

The Holt-Winters method, also known as triple exponential smoothing, is a sophisticated forecasting technique used for time series data exhibiting both trend and seasonality. This method is particularly valuable as it incorporates additional smoothing equations to account for these elements, making it ideal for more complex datasets.

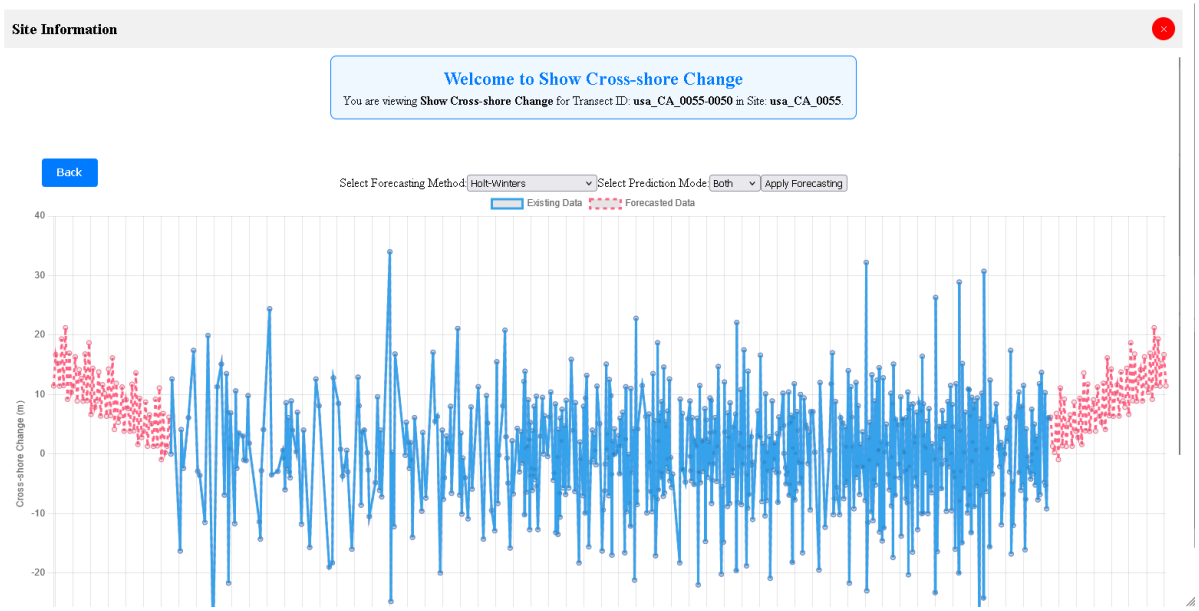

Fig41. Holt-Winters time forecasting.

The implementation of these various forecasting techniques provides a comprehensive approach to understanding and predicting shoreline changes. By analyzing past data and

projecting future trends, we can gain valuable insights into coastal dynamics, which is crucial for effective coastal management and mitigation strategies. This leads us to a broader discussion on the implications and effectiveness of these methods in real-world applications.

9. Discussion

One of the significant challenges encountered when retrieving satellite imagery is the high cloud contamination present in some images. This contamination can severely impact the ability of the CoastSat model to accurately extract shoreline data, as illustrated in the image below.

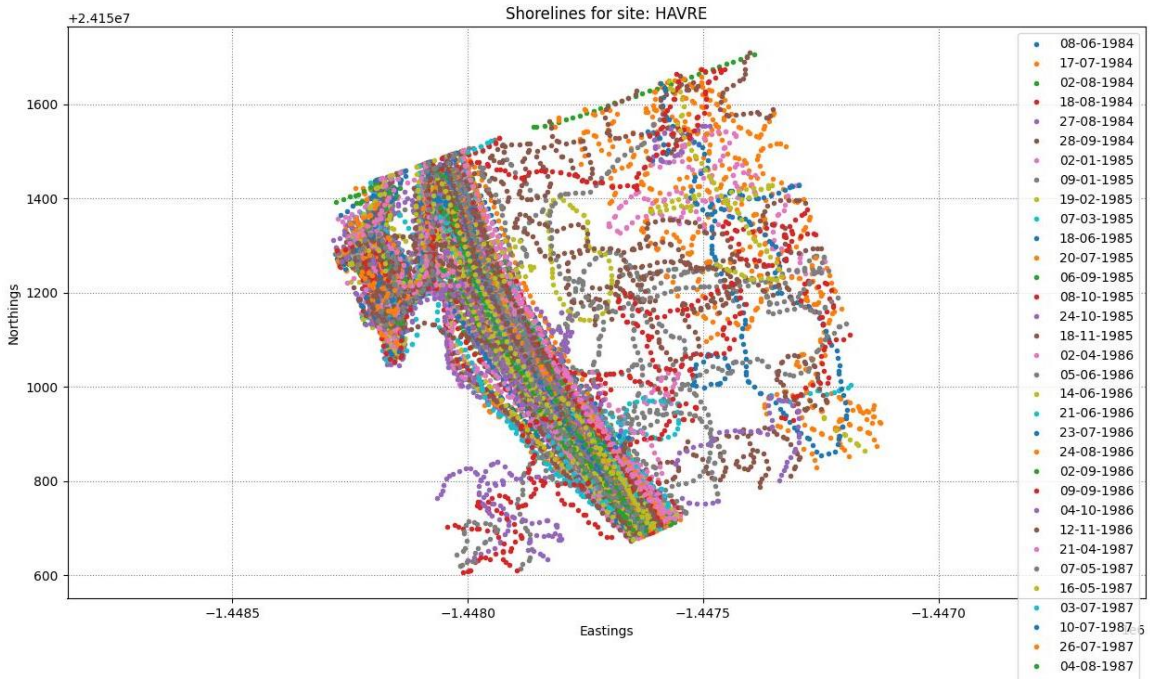

Fig42. Shoreline HAVRE all times.

For more information on addressing cloud contamination in NDVI images through SAR-Optical fusion using spatio-temporal partitioning and multiple linear regression, refer to the following article: Reconstructing Cloud-Contaminated NDVI Images. Additionally, improvements in handling cloud contamination can be explored using the resources available in this GitHub repository: DIL_SDS on GitHub.

Another critical challenge is making accurate AI predictions to forecast future or past coastlines. The implementation of time series forecasting methods, as demonstrated in this project, can provide better estimations of shoreline changes over time. By integrating advanced forecasting techniques such as the Naive method, Seasonal Naive method, and Holt-Winters method, we can gain more reliable insights into coastal dynamics and better inform coastal management strategies.

Overall, addressing these challenges requires a combination of improved data preprocessing to mitigate cloud contamination and the application of robust forecasting models to predict shoreline changes accurately. Future work should focus on enhancing these methodologies to improve the precision and applicability of shoreline predictions.

References

- Catherine Seale, Thomas Redfern, Paul Chatfield, Chunbo Luo, Kari Dempsey, Coastline detection in satellite imagery: A deep learning approach on new benchmark data, Remote Sensing of Environment, Volume 278, 2022, 113044, ISSN 0034-4257, https://doi.org/10.1016/j.rse.2022.113044. (https://www.sciencedirect.com/science/article/pii/S0034425722001584)

- Harley, M.D., Turner, I.L., Short, A.D., & Ranasinghe, R. (2011). A reevaluation of coastal embayment rotation: The dominance of cross-shore versus alongshore sediment transport processes, Collaroy–Narrabeen Beach, southeast Australia. Journal of Geophysical Research: Earth Surface, 116(F4). https://doi.org/10.1029/2011JF001989

- Luijendijk, A., Hagenaars, G., Ranasinghe, R., Baart, F., Donchyts, G., & Aarninkhof, S. (2018). The State of the World's Beaches. Scientific Reports, 8(1). https://doi.org/10.1038/s41598-018-24630-6

- Vos, K., Harley, M.D., Splinter, K.D., Turner, I.L., & Ruessink, B.G. (2019). Sub-annual to multi-decadal shoreline variability from publicly available satellite imagery. Coastal Engineering, 150, 160-174. https://doi.org/10.1016/j.coastaleng.2019.04.004

- Seale, C., Redfern, T., Chatfield, P., Luo, C., & Dempsey, K. (2022). Sentinel-2 water edges dataset (SWED): A globally distributed and labelled dataset for the development and benchmarking of techniques for automated coastline extraction. Remote Sensing of Environment, 278, 113044. https://doi.org/10.1016/j.rse.2022.113044

- Ullman, D.S., & Codiga, D.L. (2004). Seasonal variation of a coastal jet in the Long Island Sound outflow region based on HF radar and Doppler current observations. Journal of Geophysical Research: Oceans, 109(C7). https://doi.org/10.1029/2002JC001660

- Wolff, C., Schüttrumpf, H., & Ludwig, J. (2012). Development of a forecasting model to predict the morphological evolution of estuarine shorelines due to waves and tides. Coastal Engineering Proceedings, 1(33), sediment.22. https://doi.org/10.9753/icce.v33.sediment.22

- Yongjing Mao, Thomas G. Van Niel, Tim R. McVicar, Reconstructing cloud-contaminated NDVI images with SAR-Optical fusion using spatio-temporal partitioning and multiple linear regression, Remote Sensing of Environment, Volume 278, 2023, 113061, ISSN 0924-2716, https://doi.org/10.1016/j.rse.2023.113061

• Arjen Luijendijk, Gerben Hagenaars, Roshanka Ranasinghe, Fedor Baart, Gennadii Donchyts & Stefan Aarninkhof, The State of the World's Beaches, Nature Scientific Reports, 2018. https://www.nature.com/articles/s41598-018-24630-6