AUTOMATIC SHORELINE EXTRACTION TOOLBOX BASED ON MODIFIED NORMALIZED DIFFERENCE WATER INDEX OTSU THRESHOLD METHOD

Curtis Amo Dwira
MSc. Civil Engineering, Louisiana State University

Problem Statement

Shoreline monitoring is a crucial aspect of coastal management, especially in areas experiencing rapid environmental change due to natural or anthropogenic activities. Traditional shoreline extraction methods are often time-consuming, require extensive manual digitization, and are prone to subjectivity (Kermani et al., 2016). To address these challenges, this project introduces an automated shoreline extraction toolbox in ArcGIS Pro, named **Shoreline Extraction** Toolbox.

This tool leverages the Modified Normalized Difference Water Index (MNDWI) in combination with Otsu's thresholding technique to delineate water boundaries effectively. Designed with flexibility in mind, it allows the user to input Sentinel-2 composite imagery, specify band names (e.g., B3 for green and B11 for SWIR), clip to a study area, and define an output coordinate system. The extracted shoreline is saved and visualized within ArcGIS Pro, providing an end-to-end solution for coastal delineation.

This report details the development and application of an ArcGIS Pro tool designed to automate the extraction of shorelines from raster imagery using the Modified Normalized Difference Water Index (MNDWI) and Otsu's thresholding method. This tool aims to provide users with a streamlined workflow for generating shoreline data in a standard GIS format.

Objectives:

The primary objective of this report is to document the functionality, methodology, and application of the "Shoreline Extraction using MNDWI and Otsu" ArcGIS Pro tool. Specific objectives include:

i. To demonstrate the tool's output, including the extracted shoreline vector data and the MNDWI raster.

ii. To discuss the potential benefits and limitations of the tool for shoreline mapping.

Methodology

Data Source

The tool is designed to accept any georeferenced raster dataset as input, provided it contains spectral bands suitable for calculating the MNDWI (typically Green and SWIR bands). Example data sources could include:

- Landsat imagery (e.g., Landsat 8 OLI, Landsat 9 OLI-2).
- Sentinel-2 imagery.
- Other multispectral or hyperspectral imagery.

The user also defines the study area, which can be specified through a feature layer or by drawing a graphic within ArcGIS Pro. This defines the spatial extent for processing. The user specifies the desired output coordinate system, allowing for on-the-fly projection.

The tool's flexibility in accepting user-defined raster and study areas makes it adaptable to various geographic locations and data sources relevant to shoreline analysis, including potential applications in the coastal regions around Baton Rouge using available satellite imagery.

Processing Steps

*User Input***:** The user provides a raster layer, defines the study area, and specifies the output coordinate system and file paths.

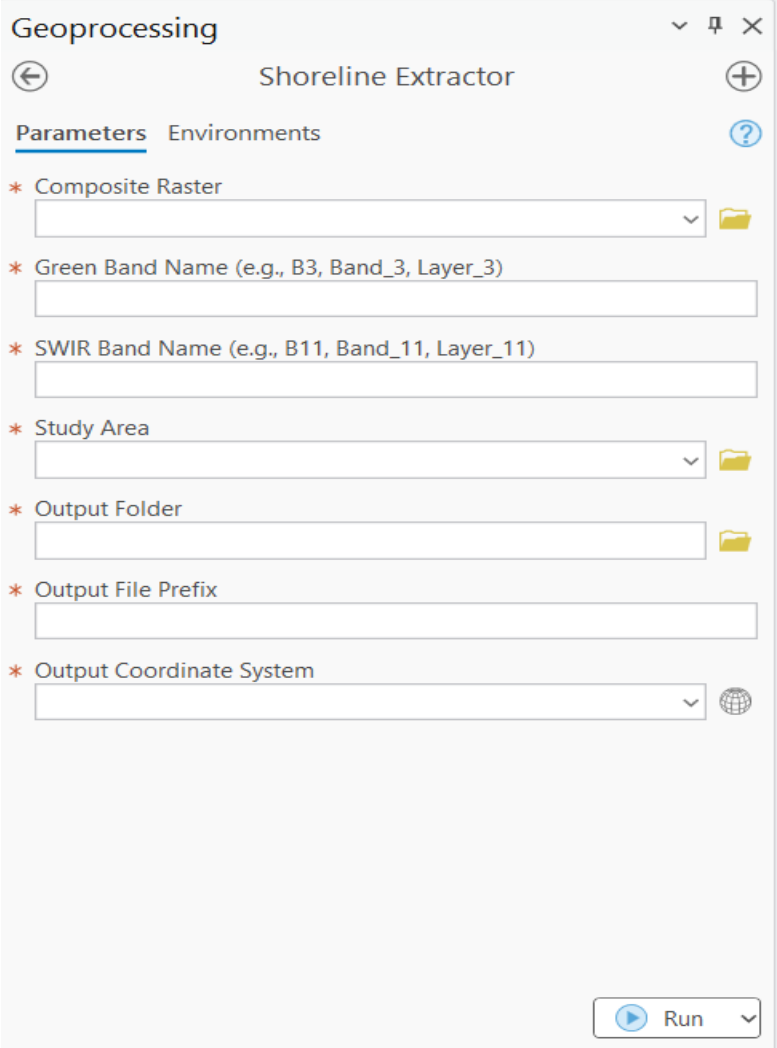

Figure 1: Toolbox Interface

MNDWI Calculation: The tool calculates the MNDWI using the formula:

$$MNDWI = (Green - SWIR) / (Green + SWIR)$$

where 'Green' and 'SWIR' represent the reflectance values of the green and shortwave infrared bands of the input raster, respectively. This index enhances water features while suppressing land and vegetation.

Otsu's Thresholding: The tool applies Otsu's method to the MNDWI raster to automatically determine an optimal threshold value (Mohamdiazer et al., 2024). Otsu's method aims to find the

threshold that minimizes the intra-class variance of the thresholded black and white pixels (water and non-water). This results in a binary raster where water pixels are assigned one value (e.g., 1) and non-water pixels another (e.g., 0).

Binary Raster Creation: A binary raster is created based on the determined Otsu threshold, effectively segmenting water from land.

Raster to Polygon Conversion: The binary raster is converted into a polygon feature class, where each contiguous group of water pixels becomes a polygon.

Polygon to Line Conversion: The polygon boundaries are extracted and converted into a polyline feature class, representing the extracted shoreline.

Coordinate System Projection: The output shoreline feature class is projected to the Geographic coordinate system specified by the user.

Output Generation: The tool saves the resulting shoreline as a shapefile and the MNDWI raster as a TIFF file in the user-defined output locations.

These steps represent the core logic implemented within your ArcGIS Pro tool, automating the process of shoreline extraction from raster data.

Analysis Approach:

The tool leverages the spectral properties of water, which typically exhibits high reflectance in the green band and low reflectance in the SWIR band. The MNDWI exploits this difference to enhance water bodies.

Otsu's method provides an unsupervised and adaptive approach to determine the optimal threshold for separating water and non-water pixels, reducing the need for manual threshold selection.

The conversion from raster to polygon and then to line simplifies the representation of the shoreline as a linear feature suitable for further GIS analysis and mapping.

Projecting the output to a user-defined coordinate system ensures compatibility with other spatial datasets and facilitates accurate spatial analysis.

Code Documentation

High-Level Code Outline

Import necessary arcpy modules

```
import arcpy
import numpy as np
from arcpy.sa import *
from skimage.filters import threshold_otsu  # Importing the Otsu thresholding
method from skimage
import os
```

Define tool parameters (input raster, study area, output paths, coordinate system)

```
        # Input values
        composite = parameters[0].value
        green_band_name = parameters[1].valueAsText
        swir_band_name = parameters[2].valueAsText
        study_area = parameters[3].value
        output_folder = parameters[4].valueAsText
        prefix = parameters[5].valueAsText
        output_crs = parameters[6].value
```

Function: Calculate MNDWI

```
        # Use Describe to get catalogPath
        raster_path = arcpy.Describe(composite).catalogPath
        green_raster = arcpy.Raster(f"{raster_path}/{green_band_name}")
        swir_raster = arcpy.Raster(f"{raster_path}/{swir_band_name}")

        green_array = arcpy.RasterToNumPyArray(green_raster)
        swir_array = arcpy.RasterToNumPyArray(swir_raster)

        # Compute MNDWI and threshold
```

```
        mndwi_array = (green_array.astype(float) - swir_array.astype(float)) /
(green_array + swir_array + 1e-10)
      # File paths
        binary_raster_path = os.path.join(output_folder, f"{prefix}_binary.tif")
        mndwi_raster_path = os.path.join(output_folder, f"{prefix}_mndwi.tif")
        polygon_path = os.path.join(output_folder, f"{prefix}_water_polygon.shp")
        valid_polygons = os.path.join(output_folder,
f"{prefix}_valid_polygons.shp")
        shoreline_raw_path = os.path.join(output_folder,
f"{prefix}_shoreline_raw.shp")
        shoreline_proj_path = os.path.join(output_folder,
f"{prefix}_shoreline_projected.shp")
```

Function: Apply Otsu threshold and Create Binary Raster

```
        threshold = threshold_otsu(mndwi_array[~np.isnan(mndwi_array)])
        binary_array = np.where(mndwi_array > threshold, 1, 0).astype(np.uint8)
```

Function: Raster to Polygon

```
        # Raster → Polygon
        arcpy.RasterToPolygon_conversion(binary_raster_path, polygon_path,
"SIMPLIFY", "Value")
        messages.addMessage("Water polygons created (with simplification).")
```

Function: Polygon to line

```
        # Filter only valid water polygons
        arcpy.MakeFeatureLayer_management(polygon_path, "poly_lyr")
        arcpy.SelectLayerByAttribute_management("poly_lyr", "NEW_SELECTION",
'"gridcode" = 1')
        arcpy.CopyFeatures_management("poly_lyr", valid_polygons)
        messages.addMessage("Filtered valid water polygons for conversion.")


        try:
```

```
            arcpy.PolygonToLine_management(valid_polygons, shoreline_raw_path)
            messages.addMessage("Shoreline (unprojected) extracted.")
        except Exception as e:
            messages.addErrorMessage("Polygon to Line conversion failed: " +
str(e))
            raise
```

Clip Input Raster to Study Area

```
        # Save MNDWI raster (clipped to study area if provided)
        ll = processed_raster.extent.lowerLeft
        cs = processed_raster.meanCellWidth
        mndwi_raster = arcpy.NumPyArrayToRaster(mndwi_array, ll, cs, cs)
        if study_area:
            mndwi_raster = ExtractByMask(mndwi_raster, projected_study_area)
        mndwi_raster.save(mndwi_raster_path)
        messages.addMessage("MNDWI raster saved.")

        # Save binary raster (clipped to study area if provided)
        binary_raster = arcpy.NumPyArrayToRaster(binary_array, ll, cs, cs,
value_to_nodata=255)
        if study_area:
            binary_raster = ExtractByMask(binary_raster, projected_study_area)
        binary_raster.save(binary_raster_path)
        messages.addMessage("Binary raster saved.")
```

Project the shoreline to the specified coordinate system

```
   # Define spatial reference for the unprojected shoreline using the output CRS
        arcpy.DefineProjection_management(shoreline_raw_path, output_crs)
        messages.addMessage(f"Spatial reference defined for: {shoreline_raw_path}
using output CRS.")

        # Project shoreline (now it's essentially projecting from the output CRS
to the output CRS, which is fine)
```

```
        arcpy.Project_management(shoreline_raw_path, shoreline_proj_path,
output_crs)
        messages.addMessage(f"Projected shoreline saved: {shoreline_proj_path}"
```

arcpy.AddMessage("Shoreline extraction complete.")

```
        try:
            aprx = arcpy.mp.ArcGISProject("CURRENT")
            map_obj = aprx.activeMap
            map_obj.addDataFromPath(mndwi_raster_path)
            map_obj.addDataFromPath(shoreline_proj_path)
            messages.addMessage("MNDWI and shoreline layers added to current
map.")
        except Exception as e:
            messages.addWarningMessage("Could not add to map: " + str(e))

        arcpy.CheckInExtension("Spatial")
        return

    def postExecute(self, parameters):
        """Just Believe."""
        return
```

Explanations of Key Functions

`calculate_mndwi (input_raster, output_raster):` This function takes the input raster path and the desired output path for the MNDWI raster. It uses the `arcpy.Raster` object to access the Green and SWIR bands (assuming they are band indices 2 and 4 respectively for Landsat 8 - this would need to be dynamic or user-configured for different sensors). It then applies the MNDWI formula using raster algebra within ArcGIS Pro and saves the resulting raster.

`apply_otsu_threshold(input_raster):` This function takes a raster object as input. It uses `arcpy.sa.OtsuThreshold()` from the Spatial Analyst module to calculate the optimal threshold value for the raster. The function returns this threshold value.

`create_binary_raster (input_raster, threshold, output_raster):` This function takes the input raster, the calculated threshold value, and the desired output path. It uses the `arcpy.sa.Con()` function to create a binary raster where pixel values above the threshold are assigned one value (e.g., 1 for water) and values below or equal are assigned to another (e.g., 0 for non-water). The resulting binary raster is then saved.

`raster_to_polygon (input raster, output polygon):` This function uses the `arcpy.RasterToPolygon_conversion ()` tool to convert the binary raster into a polygon feature class. It simplifies the polygons and creates a single-part feature for each contiguous water body.

`polygon_to_line(input_polygon,output_line):` This function uses the `arcpy.FeatureToLine_management ()` tool to extract the boundaries of the water polygons and convert them into a polyline feature class, representing the shoreline (Esri, 2025).

Results and Visualization

The primary output of the tool is a shapefile containing the extracted shoreline as a polyline feature class. The accuracy of the extracted shoreline depends on factors such as the spatial resolution of the input raster, atmospheric conditions during image acquisition, and the complexity of the coastal environment. The tool also produces an MNDWI raster as a TIFF file, which can be used to visually assess the separation between water and land and to understand the underlying spectral characteristics. The Otsu threshold value, reported during the tool execution, provides insight into the automatically determined separability of water and non-water pixels based on the MNDWI.

Geospatial practitioners could use this tool with freely available Landsat or Sentinel-2 imagery to generate up-to-date shoreline data for the Mississippi River, surrounding bayous, and coastal wetlands. The resulting shapefile can then be used for various spatial analyses within ArcGIS Pro, such as calculating shoreline change rates over time (by running the tool on imagery from different dates) or assessing the proximity of infrastructure to the shoreline. The MNDWI raster provides a visual layer to verify the accuracy of the extraction.

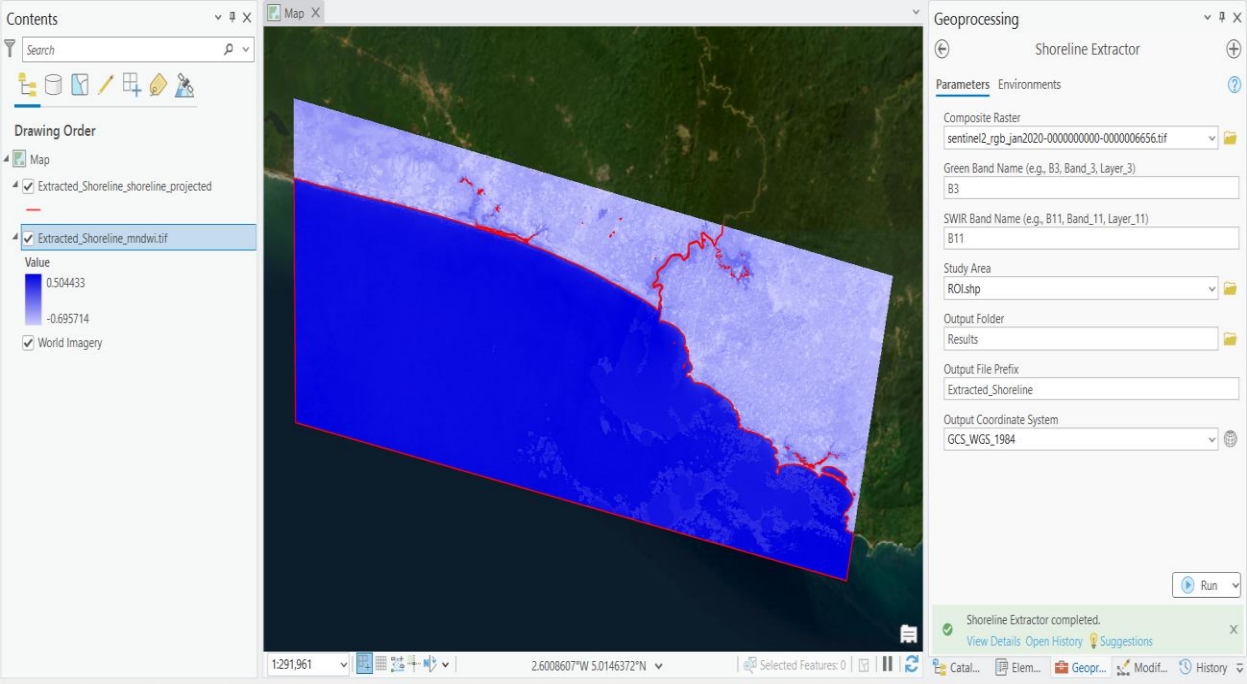

Figure 2: Final Output based on MNDWI and Otsu Threshold Method

Discussion of Limitations:

i. The accuracy of the MNDWI is sensitive to mixed pixels (pixels containing both water and land), which can occur along narrow shorelines or in areas with emergent vegetation.

ii. Otsu's method assumes a bimodal histogram (two distinct peaks representing water and non-water). In complex coastal environments with varying water turbidity or submerged vegetation, this assumption may not always hold, potentially leading to inaccurate thresholding.

iii. The tool relies on the availability of suitable spectral bands (Green and SWIR) in the input raster.

iv. The current version does not incorporate any post-processing steps to smooth the extracted shoreline or remove potential artifacts.

v. The tool's performance may vary depending on the spatial resolution and quality of the input raster.

vi. The tool best works when the data is projected in GCS_WGS_1984.

vii. When applying this tool to the dynamic coastal areas, challenges might arise in areas with significant vegetation encroachment into water bodies or in turbid waters of the river,

potentially affecting the accuracy of the MNDWI and the effectiveness of Otsu's thresholding.

Potential Improvements:

i. Implement options for different water indices (e.g., Normalized Difference Vegetation Index - NDVI for masking vegetation, other water-specific indices).
ii. Incorporate alternative thresholding methods in addition to Otsu's (e.g., manual thresholding, adaptive thresholding techniques).
iii. Add post-processing functionalities to smooth the extracted shoreline (e.g., using filtering algorithms).
iv. Develop error assessment metrics to quantify the accuracy of the extracted shoreline (if ground truth data is available).
v. Explore the integration of higher-resolution data sources (e.g., drone imagery) for more detailed shoreline mapping in smaller areas.
vi. Consider adding functionality to calculate shoreline change rates directly within the tool by processing multiple time-series images.
vii. Develop a user-friendly graphical interface within ArcGIS Pro with more interactive options and feedback.
viii. Improve the codes to give better results in different coordinate systems.

References

Esri, “Plus (Spatial Analyst),” *ArcGIS Pro Tool Reference*. [Online]. Available: https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-analyst/plus.htm. [Accessed: May 4, 2025].

Kermani, S., Boutiba, M., Guendouz, M., Guettouche, M. S., & Khelfani, D. (2016). Detection and analysis of shoreline changes using geospatial tools and automatic computation: Case of Jijelian sandy coast (East Algeria). *Ocean and Coastal Management, 132*, 46–58. https://doi.org/10.1016/j.ocecoaman.2016.08.010

Mohamadiazar, N., Ebrahimian, A., & Hosseiny, H. (2024). Integrating deep learning, satellite image processing, and spatial-temporal analysis for urban flood prediction. *Journal of Hydrology, 639*, 131508. https://doi.org/10.1016/j.jhydrol.2024.131508

Gemini AI language model by Google was used for writing part of the code.