Readme

Introduction

Background
An extreme sports holiday company wants to find some extreme gradients where they might potentially set up some sporting activities. A 2D raster data set of elevations above sea level. Was given.

This is a GUI application for loading DEM data files and calculating their maximum slope and aspect. The program can display images of input terrain data, maximum slope data, slope angle data, and aspect data, and allows the user to save output images.

Dependencies

To use this application, you will need to have Python 3 and the following Python libraries installed:
- NumPy
- Matplotlib
- Tkinter

Data description

Slope.txt: a 2D raster data set of elevations above sea level.
slope.txt_max_slopes.txt: Data files generated after calculation
slope.txt_aspect.txt: Data files generated after calculation
main.py: The entry point of the application that initializes the GUI.
gui.py: Contains the code for the graphical user interface using the Tkinter library.
data_processing.py: Contains the code for calculating the maximum slope and aspect for each pixel.
data_display.py: Contains the code for displaying the raster data and saving the current image.

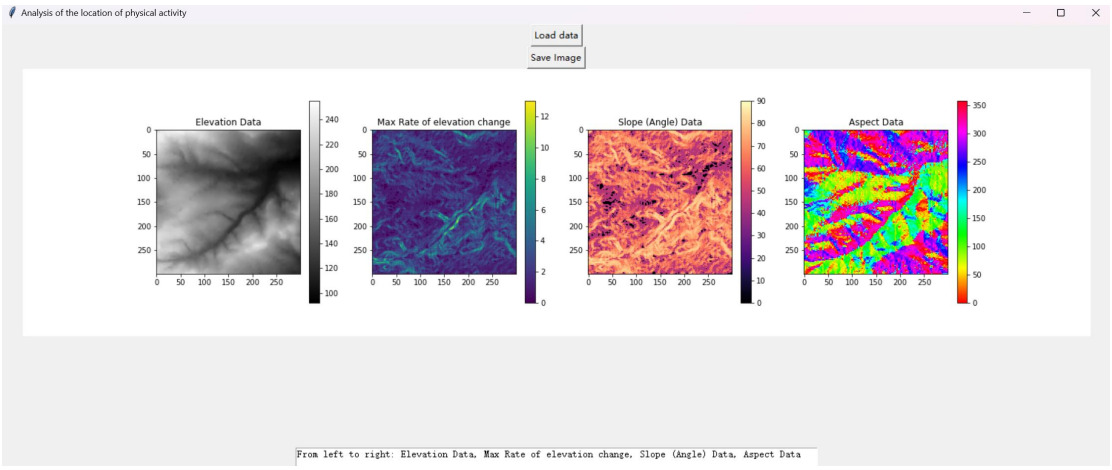
How to use

1. Clone or download the project from the Github repository.
2. Install the required libraries (NumPy, Matplotlib, Tkinter).
3. Run the `main.py` file.

Once the application runs, you can click the "Load data" button to select a text file containing raster data. The application will then calculate the maximum slope and aspect data for the raster data and display the results in four plots:

1. Elevation Data
2. Max Slope Data
3. Slope Angle Data
4. Aspect Data

You can also save the plots as image files by clicking on the "Save Image" button.



Interface example

File Structure

gui.py

This code implements a GUI application for loading the elevation data file, calculating its maximum slope and aspect, and displaying four images in the GUI interface: the elevation data image, the Max Rate of elevation change image, the slope angle image, and the slope to the image. At the same time, the user can save the displayed four images as image files in PNG or JPEG format by clicking the "Save Image" button.

First, the program defines a class called "GUI", which contains three data variables: elevations, max_slopes, and aspect. These three variables are used to store the elevation data read from the elevation data file, the calculated Max Rate of elevation change and aspect respectively. Then, the program creates two buttons, the "Load Data" button and the "Save Image" button, and binds them to the corresponding functions. Among them, the state of the "save image" button is set to "disabled", because the image cannot be saved before the data is loaded. Next, the program defines two functions: load_data() and save_fig(). The load_data() function is used to load the elevation data file, calculate the maximum slope and aspect, save the calculation results to a file and display four images in the GUI interface. The save_fig() function is used to save four images as image files in PNG or JPEG format.

The specific implementation method is: in the load_data() function, the program first uses the file dialog module in the tkinter module to open a file selection dialog box, allowing the user to select an elevation data file. Then, the program reads the elevation data from the selected file using the loadtxt() function in the numpy module, and calculates the maximum slope and aspect using the calculate_slope() and calculate_aspect() functions. The calculation results are saved to a file and four images are displayed in the GUI interface using the display_data() function. Finally, the state of the "Save Image" button is set to "Enabled", and the image can be saved.

In the save_fig() function, the program first uses the filedialog module in the tkinter module to open a file saving dialog box, allowing the user to select a file name and save location. Then, the program calls the save_fig() function in the display_data module to save the four images as image files in PNG or JPEG format.

data_processing.py

This code includes two functions to calculate the Max Rate of elevation change and aspect, respectively.
The maximum slope actually calculates the Max Rate of elevation change, but for the convenience of reference and can be expressed as an angle that can convert the Rate of elevation change into a slope, the variables here are all declared as the maximum slope

The implementation of the calculate_slope function is:

First, the input data is padded in edge mode to handle edge cases.

Then initialize an empty max_slope list for storing the maximum slope of each pixel.

For each pixel, firstly obtain its current height, then traverse its 8 adjacent pixels, and calculate the height difference from the current pixel.

Record the maximum value of the height difference between all adjacent pixels and the current pixel, and then calculate the slope value between this pixel and its adjacent pixels.

Store the maximum slope value for this pixel in the max_slope list and return the list.

The implementation of the calculate_aspect function is:

First, the input data is padding in edge mode to handle edge cases.

Then initialize an empty aspect list to store the aspect of each pixel.

For each pixel, calculate the height difference between its horizontal and vertical directions and its slope.

By calculating the angle between the slope and the horizontal plane, and converting the angle to a value within 360 degrees, the aspect of the pixel is obtained.

Store the aspect value of this pixel into the aspect list and return the list.

These two functions implement the function of converting elevation data into slope and aspect data, which is the basis for making topographic maps and terrain analysis. Among them, the use of numpy's pad function to pad the data is to handle edge cases. When traversing each pixel, use two layers of loops to traverse its adjacent pixels, calculate the slope by calculating the height difference between each adjacent pixel and the current pixel, and calculate the slope by calculating the angle between the slope and the horizontal plane Towards. Finally, store the result of the calculation in the corresponding list, and return the result

data_display.py

This code mainly defines a function display_data() for displaying terrain data on the GUI interface. The following are the specific implementation details of this code:

1. Import required modules and libraries, including numpy, matplotlib.pyplot and tkinter, etc.

2. Define the display_data() function, which contains four input parameters, namely terrain data (elevations), maximum slope data (max_slopes), slope direction data (aspect) and Tkinter window (window).

3. Obtain the number of rows and columns of the terrain data matrix.

4. Create a Matplotlib graphics object that contains four subgraphs for displaying terrain data, maximum slope data, slope direction data, and data description.

5. Display the terrain data in the first subgraph, use the imshow() function to draw the image, set the colormap to "gray", set the interpolation method to "nearest" and display the colorbar.

6. Display the Max Rate of elevation change in the second subgraph, use the imshow() function to draw the image, set the colormap to "viridis", set the interpolation method to "nearest" and display the colorbar.

7. Calculate the slope angle of each pixel for display in the third sub-image, use the arctan() function to calculate the angle, multiply it by 180/π to convert radians into angles, and use the imshow() function to draw the image, set Set the colormap to "magma", set the interpolation method to "nearest", set the vmin and vmax parameters to 0 and 90, which represent the lower limit and upper limit of the slope angle range, and display the colorbar.

8. Display the slope direction data in the fourth subgraph, use the imshow() function to draw the image, set the colormap to "hsv", set the interpolation method to "nearest" and display the colorbar.

9. Convert the Matplotlib graphics object to a Tkinter-compatible image format and embed it in a Tkinter window.

10. Add a text box at the bottom of the Tkinter window to display data description text.

11. Disable the editing function of the text box.

12. Return the result.

The overall idea of this code is to use Matplotlib to draw four subgraphs to display terrain data, Max Rate of elevation change, slope direction data and data description, and convert the drawn results into visualized images in the Tkinter window. When displaying data, different colormaps are used to distinguish different data types and a color bar is added to the right of each subgraph. Finally, a text box is added to display data captions to give users a better understanding of the data.

test.py

This is a Python unit test script used to test two functions, calculate_slope and calculate_aspect, in terrain data processing code. The purpose of unit testing is to ensure the correctness and reliability of the code under different input conditions.

Specifically, the script imports the unittest and numpy libraries and defines a TestSlopeAspect test class. The class contains two test functions: test_calculate_slope and test_calculate_aspect. The setUp function runs before each test function and initializes the test data and cell size.

The test_calculate_slope and test_calculate_aspect functions test whether the calculate_slope and calculate_aspect functions correctly calculate the rate of elevation change and the aspect of the terrain data. The functions first call the two functions, passing the test data and cell size as arguments. It then compares the function's return value with the expected results and checks if they are equal using the assertTrue function. If the expected results differ from the actual results, error messages are printed to aid in debugging.

```
In [1]: runfile('E:/leeds_2023/GEOG5990M_Programming_for_GIS/assignment2/优化3/test.py', wdir='E:/leeds_2023/
GEOG5990M_Programming_for_GIS/assignment2/优化3')
..
----------------------------------------------------------------------
Ran 2 tests in 0.005s

OK
```

Test result

License

This project is licensed under the MIT License.