**Boston University**

**Electrical & Computer Engineering**

**EC463 Senior Design Project**

First Prototype Test Report

**Coastline Prediction using Existing Climate Change Models**

By

Team 18

Team Sea Rise



Team Members

Beatriz Sicilia [bsicilia@bu.edu](mailto:email1@bu.edu)

Stacia Kolodziejski [skolodz@bu.edu](mailto:email1@bu.edu)

Wenyu (Jessica) Hu [wjhu@bu.edu](mailto:email1@bu.edu)

Saif Alblooshi [sjalbloo@bu.edu](mailto:email1@bu.edu)

**Introduction**

The set up for this test is pretty simple as the only required materials are a computer, MatLab, and the elevation data files. Once the computer is set up, open MatLab and locate the data file of interest (elevation data of the specific region). Then, run the MatLab script to accurately depict the data of the specific region onto a showing the corresponding elevation.

We ran two different scripts, one of them using .laz data and the other one using .tif data.

**Data formats**

**About*.tif* Files**

A TIFF, Tagged iImage File Format, is a file where an image is saved. A TIFF has many functions, one of which is that it is used to store high quality images with many colors and usually digital photos. *.tif* files can be opened using different image editors (EX: Microsoft Photos, ACD system, ACDsee photo studio) as many people in the graphics industry use *.tif* files.

**About *.laz* Files**

A LAZ file is a compressed lidar data file. It is easier to share the *.laz* file as it is less than 20% of the original *.las* file size. Plus, it helps with storing the data and the file. A .las file needs to be compressed, because it contains data of the Earth's surface using LIDAR, a light detection and ranging technology, which are usually very heavy and dense. Therefore compressing them is the more efficient way to share and store these data.

**Process**

We started with the **.tif** files. Out of the multiple functions we tried, we finally decided to use the following code:

t1 = Tiff('USGS\_13\_n43w071\_20220713.tif','r');

imageData1 = read(t1);

Tiff is a built-in MatLab function that identifies the file. This way, the *read* function will know “what” it is reading and how to put the read data into a matrix.



**Figure 1** shows the output of the Tiff function.

**Figure 2** shows the matrix that the Tiff function helped to read into the imageData1 variable.



The 10812 x 10812 matrix in *Figure 2* defines the elevation data of the area that the “USGS\_13\_n43w071\_20220713.tif” file covers on a map. Once we have the data in this format, we can display it using the following code:

figure

imshow(imageData1, [-2 maxNum1]);

colorbar

The [-2 maxNum1] allows the display to be grayscaled based on the elevation data (See Figure 4 & 5).

Now that we can display one area of the map from a TIFF, we can do the same for multiple files, which will allow us to create the display for a larger area of the map. We decided to concatenate the matrices and display it using the same code imshow. Figure 3 shows the large 21624 x 10812 matrix.

****

Once we understood how to operate with the *.tif* files, we tried *.laz* files.

(See final results in the ***Outputs*** section.)

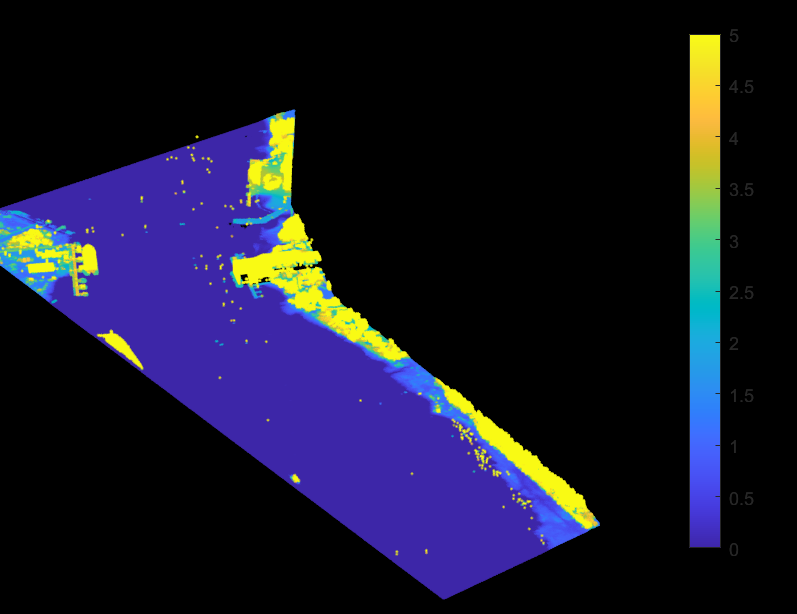
**Output**

**Of*.tif* Files**





**Of*.laz* Files**



**Matlab script**

**Of*.tif* Files**

%% n43w071

t1 = Tiff('USGS\_13\_n43w071\_20220713.tif','r');

imageData1 = read(t1);

maxNum1 = matmax(imageData1);

figure

imshow(imageData1, [-2 maxNum1]);

colorbar

%% n43w072

t2 = Tiff('USGS\_13\_n42w071\_20191216.tif','r');

imageData2 = read(t2);

maxNum2 = matmax(imageData2);

figure

imshow(imageData2, [-2 maxNum2]);

colorbar

%% Concatenate Matrices

combineData = [imageData1; imageData2];

figure

imshow(combineData, [-2 maxNum2]);

colorbar

%% Functions

function maxNum = matmax(matrix)

maxarr = max(matrix);

maxNum = max(maxarr);

end

function minNum = matmin(matrix)

minarr = min(matrix);

minNum = min(minarr);

end

**Of *.laz* Files**

path = fullfile("data.laz");

lasReader = lasFileReader(path);

ptCloud = readPointCloud(lasReader);

figure

pcshow(ptCloud.Location)

clim([0 10])

**Conclusion**

Over all, we downloaded the files for the elevation data of the region we would like to use our function in and visualize the rising sea level for, after selecting the region and downloading the file we read the files and we used matlab to convert and open these files so we can compare and download the rising sea level data so that we can start with writing our function.