# Applying Average Similar Neighborhood Filter to Satellite Images in Arcpy

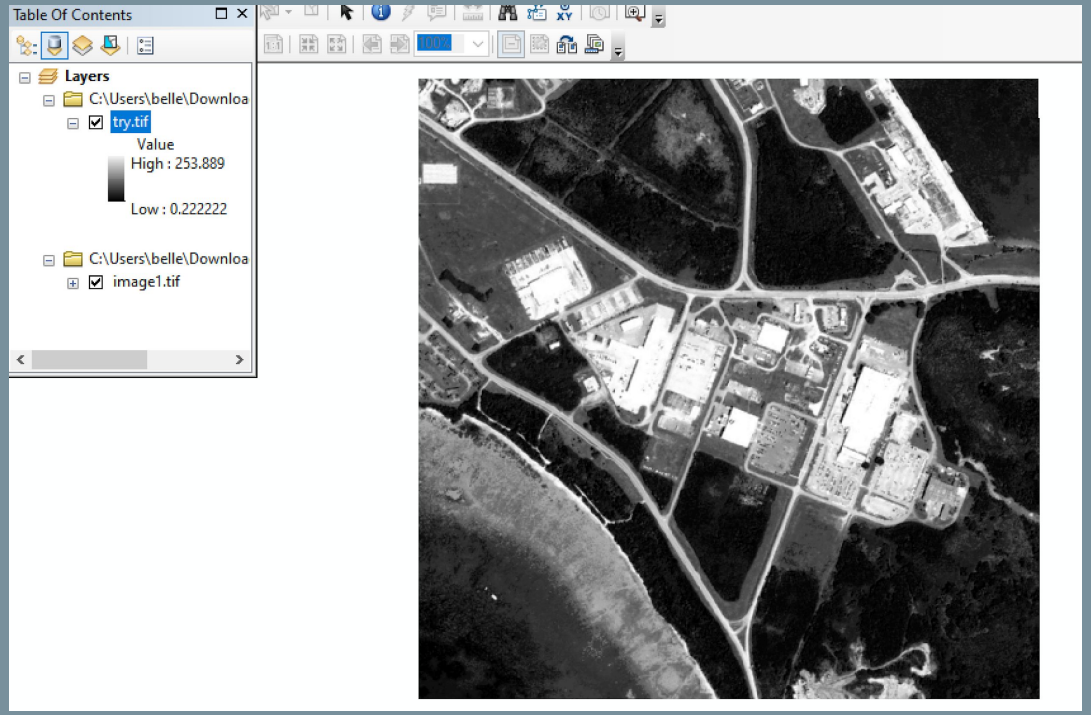
## 2019 Geospatial Software Design Assignment 10

Zixi Liu

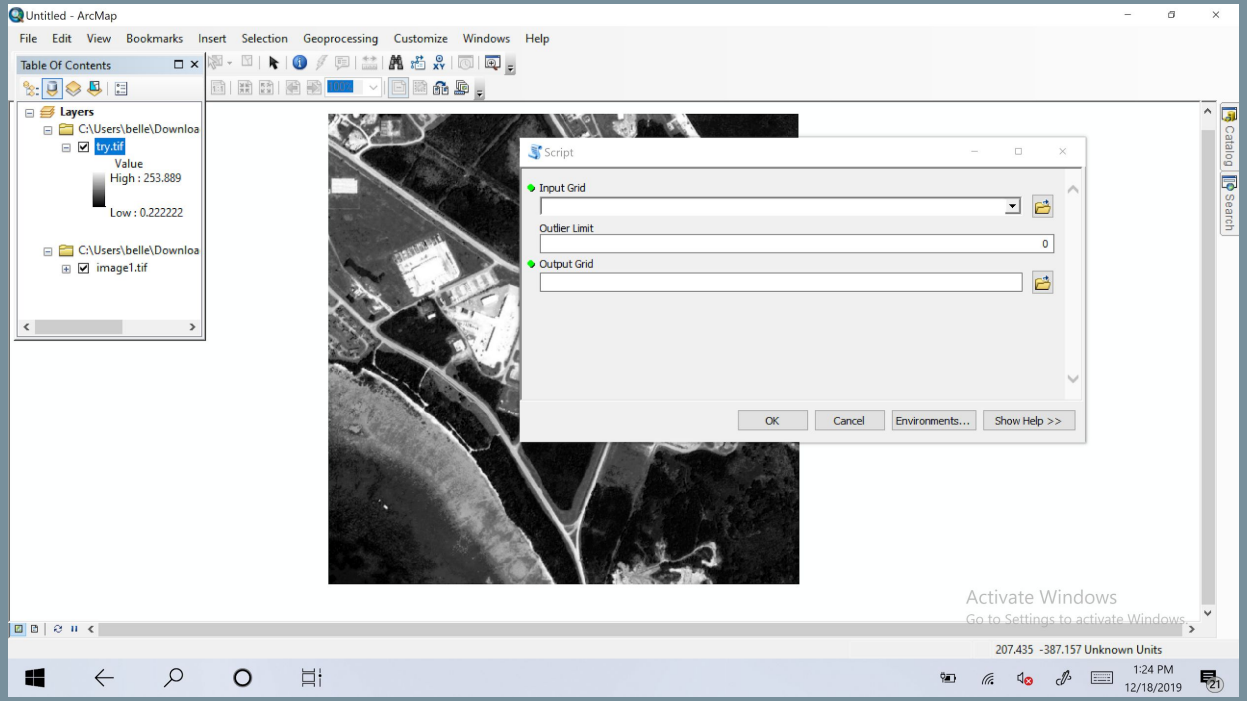
Nov, 2019

# Overview: Reduce Noise by Averaging Similar Neighborhoods

- On the left is the Tiled High Resolution Orthoimagery for Guam; on the left is the image after applying our arctool of averaging similar neighborhood pixels. We are able to reduce outliers in the pixels.



# Overview: What Our Arctoolbox Looks Like




- ➔ Input Grid is a raster layer.
- ➔ Outlier Limit is a double with a default of 0.
- ➔ Output Grid is a raster dataset.
- ➔ This filter is used for image processing before edge detection.

# Understanding the Python Script

Grid.py - C:/Users/belle/Downloads/CPLN670/ACS_Tracts/Grid.py (2.7.16)

File Edit Format Run Options Window Help

```
"""
THIS SCRIPT ASSIGNS TO EACH PIXEL THE MEAN OF ALL VALUES WITHIN ITS IMMEDIATE NEIGHBORHOOD
EXCEPT FOR THOSE THAT ARE DEEMED TO BE OUTLIERS, WHERE AN OUTLIER IS A VALUE ABOVE OR BELOW
A SPECIFIED NUMBER OF STANDARD DEVIATIONS FROM THE MEAN OF ALL IMMEDIATELY NEIGHBORING VALUES.

To create an ArcToolbox tool with which to execute this script, do the following.
1   In  ArcMap > Catalog > Toolboxes > My Toolboxes, either select an existing toolbox
    or right-click on My Toolboxes and use New > Toolbox to create (then rename) a new one.
2   Drag (or use ArcToolbox > Add Toolbox to add) this toolbox to ArcToolbox.
3   Right-click on the toolbox in ArcToolbox, and use Add > Script to open a dialog box.
4   In this Add Script dialog box, use Label to name the tool being created, and press Next.
5   In a new dialog box, browse to the .py file to be invoked by this tool, and press Next.
6   In the next dialog box, specify the following inputs (using dropdown menus wherever possible)
    before pressing OK or Finish.
        DISPLAY NAME        DATA TYPE       PROPERTY>DIRECTION>VALUE    DEFAULT
        Input Grid          Raster Layer    Input
        Outlier Limit       Double          Input                       0.0
        Output Grid         Raster Dataset  Output

   To later revise any of this, right-click to the tool's name and select Properties.
"""
# Import external modules
import sys, os, string, math, arcpy, traceback, numpy

# Allow output to overwite any existing grid of the same name
arcpy.env.overwriteOutput = True

# If Spatial Analyst license is available, check it out
if arcpy.CheckExtension("spatial") == "Available":
    arcpy.CheckOutExtension("spatial")

try:
   # Create a real-valued InputArray from the initial input grid and note its dimensions
   InputGridName       = arcpy.GetParameterAsText(0)
   InputArray          = arcpy.RasterToNumPyArray(InputGridName)
   InputArray          = InputArray.astype(float)
   HowManyRows         = InputArray.shape[0]
   HowManyColumns      = InputArray.shape[1]
```



Ln: 98 Col: 0

# Understanding the Python Script

Grid.py - C:/Users/belle/Downloads/CPLN670/ACS_Tracts/Grid.py (2.7.16)

File Edit Format Run Options Window Help

```
howManyColumns      = inputArray.shape[1]

# Initialize an OutputArray that is similar to that InutArray but filled with zeroes
OutputArray         = numpy.zeros_like(InputArray)

 # Get number of standard deviations to be used in defining outliers
Limit               = float(arcpy.GetParameterAsText(1))

    # Initialize vertical and horizontal offests for the nine pixels within each neighborhood
RowShift            = [0,-1, 0, 1, 0,-1,-1, 1, 1]
ColumnShift         = [0, 0, 1, 0,-1,-1, 1, 1,-1]

    # Loop through rows and columns of pixels
for ThisRow in range(HowManyRows):
        for ThisColumn in range(HowManyColumns):

            HowManyNeighbors                    = 0
            # Loop through the nine pixels in each neighborhood to compute their count and sum
            for NextNeighbor in range(9):
                NeighborRow         = ThisRow + RowShift[NextNeighbor]
                if NeighborRow      < 0 or NeighborRow      >= HowManyRows:    continue
                NeighborColumn      = ThisColumn + ColumnShift[NextNeighbor]
                if NeighborColumn   < 0 or NeighborColumn   >= HowManyColumns: continue
                # Increment neighborhood sum and neighbor count
                OutputArray[ThisRow][ThisColumn] = OutputArray[ThisRow][ThisColumn] + InputArray[NeighborRow][NeighborColumn]
                HowManyNeighbors = HowManyNeighbors + 1
            # Divide neighborhood sum by neighbor count to get mean of all pixels in neighborhood
            MeanOfAllNeighbors = OutputArray[ThisRow][ThisColumn] / HowManyNeighbors
            OutputArray[ThisRow][ThisColumn] = 0

            # Loop through the nine pixels in each neighborhood to compute their standard deviation
            for NextNeighbor in range(9):
                NeighborRow         = ThisRow + RowShift[NextNeighbor]
                if NeighborRow      < 0 or NeighborRow      >= HowManyRows:    continue
                NeighborColumn      = ThisColumn + ColumnShift[NextNeighbor]
                if NeighborColumn   < 0 or NeighborColumn   >= HowManyColumns: continue
                # Increment neighborhood sum of squared deviations
                DeviationOfThisNeighbor          = InputArray[NeighborRow][NeighborColumn] - MeanOfAllNeighbors
                SquaredDeviationOfThisNeighbor   = DeviationOfThisNeighbor * DeviationOfThisNeighbor
```



Ln: 98 Col: 0

# Understanding the Python Script

Grid.py - C:/Users/belle/Downloads/CPLN670/ACS_Tracts/Grid.py (2.7.16)

File Edit Format Run Options Window Help

```
                OutputArray[ThisRow][ThisColumn] = 0

                # Loop through the nine pixels in each neighborhood to compute their standard deviation
                for NextNeighbor in range(9):
                    NeighborRow          = ThisRow + RowShift[NextNeighbor]
                    if NeighborRow       < 0 or NeighborRow       >= HowManyRows:    continue
                    NeighborColumn       = ThisColumn + ColumnShift[NextNeighbor]
                    if NeighborColumn    < 0 or NeighborColumn    >= HowManyColumns: continue
                    # Increment neighborhood sum of squared deviations
                    DeviationOfThisNeighbor          = InputArray[NeighborRow][NeighborColumn] - MeanOfAllNeighbors
                    SquaredDeviationOfThisNeighbor   = DeviationOfThisNeighbor * DeviationOfThisNeighbor
                    OutputArray[ThisRow][ThisColumn] = OutputArray[ThisRow][ThisColumn] + SquaredDeviationOfThisNeighbor
                # Divide neighborhood sum by neighbor count to get mean squared deviation of all pixels in neighborhood
                MeanSquaredDeviationOfAllNeighbors   = OutputArray[ThisRow][ThisColumn] / HowManyNeighbors
                StandardDeviation                    = math.sqrt(MeanSquaredDeviationOfAllNeighbors)
                OutputArray[ThisRow][ThisColumn]     = 0

                HowManyNeighbors                     = 0
                # Loop through the non-outlier pixels in each neighborhood to compute their count and sum
                for NextNeighbor in range(9):
                    NeighborRow          = ThisRow + RowShift[NextNeighbor]
                    if NeighborRow       < 0 or NeighborRow       >= HowManyRows:    continue
                    NeighborColumn       = ThisColumn + ColumnShift[NextNeighbor]
                    if NeighborColumn    < 0 or NeighborColumn    >= HowManyColumns: continue
                    # Increment neighborhood sum and neighbor count
                    OutlierLimit = StandardDeviation * Limit
                    if InputArray[NeighborRow][NeighborColumn] > (MeanOfAllNeighbors + OutlierLimit): continue
                    if InputArray[NeighborRow][NeighborColumn] < (MeanOfAllNeighbors - OutlierLimit): continue
                    OutputArray[ThisRow][ThisColumn] = OutputArray[ThisRow][ThisColumn] + InputArray[NeighborRow][NeighborColumn]
                    HowManyNeighbors = HowManyNeighbors + 1
                # Divide neighborhood sum by neighbor count to get mean of all pixels in neighborhood
                OutputArray[ThisRow][ThisColumn]  = OutputArray[ThisRow][ThisColumn] / HowManyNeighbors

    # Create output grid from that new array
InputGrid           = arcpy.Raster(InputGridName)
gridExtent          = InputGrid.extent
lowerleftPoint      = gridExtent.lowerLeft
gridResolution      = InputGrid.meanCellWidth
```



Ln: 98 Col: 0

# Understanding the Python Script

Grid.py - C:/Users/belle/Downloads/CPLN670/ACS_Tracts/Grid.py (2.7.16)

File Edit Format Run Options Window Help

```
                OutputArray[ThisRow][ThisColumn]     = 0

                HowManyNeighbors                     = 0
                # Loop through the non-outlier pixels in each neighborhood to compute their count and sum
                for NextNeighbor in range(9):
                    NeighborRow         = ThisRow + RowShift[NextNeighbor]
                    if NeighborRow      < 0 or NeighborRow      >= HowManyRows:    continue
                    NeighborColumn      = ThisColumn + ColumnShift[NextNeighbor]
                    if NeighborColumn   < 0 or NeighborColumn   >= HowManyColumns: continue
                    # Increment neighborhood sum and neighbor count
                    OutlierLimit = StandardDeviation * Limit
                    if InputArray[NeighborRow][NeighborColumn] > (MeanOfAllNeighbors + OutlierLimit): continue
                    if InputArray[NeighborRow][NeighborColumn] < (MeanOfAllNeighbors - OutlierLimit): continue
                    OutputArray[ThisRow][ThisColumn] = OutputArray[ThisRow][ThisColumn] + InputArray[NeighborRow][NeighborColumn]
                    HowManyNeighbors = HowManyNeighbors + 1
                # Divide neighborhood sum by neighbor count to get mean of all pixels in neighborhood
                OutputArray[ThisRow][ThisColumn]  = OutputArray[ThisRow][ThisColumn] / HowManyNeighbors

        # Create output grid from that new array
    InputGrid           = arcpy.Raster(InputGridName)
    gridExtent          = InputGrid.extent
    lowerleftPoint      = gridExtent.lowerLeft
    gridResolution      = InputGrid.meanCellWidth
    outputGrid       = arcpy.NumPyArrayToRaster(OutputArray,lowerleftPoint,gridResolution)
    outputGrid.save(arcpy.GetParameterAsText(2))

except Exception as e:
        # If unsuccessful, end gracefully by indicating why
        arcpy.AddError('\n' + "Script failed because: \t\t" + e.message )
        # ... and where
        exceptionreport = sys.exc_info()[2]
        fullermessage   = traceback.format_tb(exceptionreport)[0]
        arcpy.AddError("at this location: \n\n" + fullermessage + "\n")

    # Check in Spatial Analyst extension license
arcpy.CheckInExtension("spatial")
```

Ln: 98 Col: 0



1:24 PM
12/18/2019