Version: 1.0
6. December 2023

Preface
This document is intended for use by satellite imagery professions in tandem with the stereo-pair retro-generator program that this document describes. This document is made for Version 1.0 and should not be used to reference future versions. Future versions are expected as the industry changes. For instance, a new version would be released if the program needed to support a new file type.

Lead Author: Scott Shlanta
Lead Author Signature:

SShlanta

Review Names: Khaliu Olonbayar, Peter Kahnke, Matthew Isbell
Review Name Signatures:

[illegible]

[illegible]

[illegible]

Version: 1.0
6. December 2023

Introduction

Satellite imagery can be used to generate images of a landscape from different perspectives. Because these photos are taken from different viewpoints, there are slight differences between the two photos. The disparity between the features in each photo can be used to calculate the depth of that feature using a process similar to the one our brains use to give depth to our vision. This process is well established in the satellite imaging industry. This program, however, is dedicated to the reversal of this process. The stereo pair retro-generator uses an ortho-corrected image and a height field to generate an entirely new "flat" image with no sense of depth. This new image is essentially one half of the original stereo pair. While it may seem futile to generate half of the stereo pair that you would have needed to create the ortho-corrected image and height field in the first place, there is actually a great deal of new information to be learned. The stereo pair retro-generator includes a baseline distance parameter, so instead of being limited to the two original viewpoints of the stereo pair, you can use the stereo pair retro-generator to change the distance between those two viewpoint, so you essentially gain the ability to see this landscape from any perspective.

Lead Author: Scott Shlanta
Lead Author Signature:
[illegible]

Review Names: Khaliu Olonbayar, Peter Kahnke, Matthew Isbell
Review Name Signatures:

[illegible]

[illegible]

[illegible]

Version: 1.0
6. December 2023

Glossary
This section defines all technical terms. It can be used for reference as one reads the document, but it is recommended that this document is read with a working understanding of these terms.

Lead Author: Scott Shlanta
Lead Author Signature:

[illegible]

Review Names: Khaliu Olonbayar, Peter Kahnke, Matthew Isbell
Review Name Signatures:

[illegible]

[illegible]

[illegible]

Glossary

Baseline Distance	The distance between the viewpoints from which the two images of a stereo pair are taken. It is analogous to the distance between our eyeballs.
De Marzio Equation	The equation relating depth and disparity in stereography. Can be used in either direction.
Depth	How far from the camera a certain feature in an image is.
Depth Field	A representation of the depth of each pixel in an image. Often represented as a black and white image in which brightness represents depth.
Disparity	The difference in location of a certain feature between the images of a stereo pair.
Distortion	The amount that a feature in an image moves when it is made into an ortho-corrected image or turned back into half of a stereo pair
Ortho-corrected image	An image made from a stereo pair and a depth field in which each feature in the image is slightly adjusted so that it more accurately depicts its depth. It's an image that approximates 3D.
Stereo pair	Two images of the same thing that were taken from a slightly different viewpoint. The disparity between where each feature is located on each image can be used to calculate depth.

Version: 1.0
6. December 2023

User Requirements Definition
This program is used to create half of a stereo pair from an ortho-corrected image and a height field. As such, it is required to be able to load and process an ortho-corrected image and a height field, preferably in the .tiff file type. It should also be able to accept the preferred baseline distance for the new half stereo pair as a parameter. This baseline distance will determine the disparity between features in the new half stereo pair and the ortho-corrected image based on the depth of the feature. The inputs to the program should be validated before they are processed to make sure that the two images that are submitted are the same size.

This product is made for the satellite imagery professional and should be usable without commercial equipment. The hardware requirements shall not exceed that of a laptop so that the program may remain accessible. The program shall generate the resulting image in an amount of time acceptable to the user so that they do not have to wait an amount of time that would impede work. The amount of time that each image takes to generate will be proportional to the size of the image, however. The user interface need not be overdesigned. The analysts using this product will have specialized industry knowledge and can be trained to use an intuitive user interface such as command line.

Lead Author: Scott Shlanta
Lead Author Signature:

SShlanta

Review Names: Khaliu Olonbayar, Peter Kahnke, Matthew Isbell
Review Name Signatures:

[illegible]

[illegible]

[illegible]

Augustana University
2001 S. Summit Ave
57197 Sioux Falls, SD

Version: 1.0
6. December 2023

System Architecture
System Architecture will present a high-level overview of the system architecture, showing the reader how the functions will be distributed across system modules.

Lead Author: Matthew Isbell
Lead Author Signature:

[illegible]

Review Names: Khaliu Olonbayar, Peter Kahnke, Scott Shlanta
Review Name Signatures:

[illegible]

[illegible]

[illegible]

Augustana University
2001 S. Summit Ave
57197 Sioux Falls, SD

This program consists of two classes, with each class containing multiple functions. The class that creates the image is known internally simply as "Forward".

Forward contains many functions that all work together to accomplish the goal of creating a flat image from an orthorectified image and a height map. While it is important to note that the names of these functions are subject to change, this is the final layout of how these functions will interact with one another.

- + get_img_input_start
    - This function takes the users string input for where the desired image is located on the computer
- + find_image
    - This function will actually load said image into the program
- + convert_image
    - This function converts the image into a 2d byte array
- + get_image_size
    - This function looks at the dimensions of te 2d array above to see the image size
- + get_rgb_value
    - This function looks at each pixel in the 2d array and looks at what the RGB color values are (Ex: 156, 20, 76)
- + get_height_map
    - Like the function that finds the image on the computer, this function will find the height map on the computer, depending on where the user says it is
- + find_height_map
    - This function loads the heightmap into the program
- + convert_height_map
    - Function convers the height map into a 2d byte array
- + get_height_map_size
    - By looking at the 2d byte array, this function determines the image dimensions
- + get_height_value
    - This function looks at each pixel in the 2d byte array and reports how bright each pixel is
- + deMarzioEquationForward
    - This function takes in everything found so far in the previous functions, and outputs a flat image in the desired output location

This is every function in the Forward class. It is important to note that our other class, internally named "Reverse", uses every single one of these functions in its own class. The only difference is the final equation, where instead of going forward, we are going backwards. In other words, instead of creating a 2D image, we are now creating a 3D image to verify whether our output was correct. This is the description of the one unique function:

- + deMarzioEquationReverse
    - This function takes the flat image generated by the forward class and a height map, and creates a 3D image to test whether our output was correct.

Version: 1.0
6. December 2023

System Requirements Specification
The System Requirements Specifications section will go into depth
on both the hardware and software requirements of the created program.

Lead Author: Matthew Isbell
Lead Author Signature:

[illegible]

Review Names: Khaliu Olonbayar, Peter Kahnke, Scott Shlanta
Review Name Signatures:

[illegible]

[illegible]

[illegible]

Augustana University
2001 S. Summit Ave
57197 Sioux Falls, SD

Functional Requirements

R1. The system shall accept and process an ortho-corrected image.

- R1.1. The system shall locate the image file in a specific input folder.
- R1.2. The system shall read in the image file
- R1.3. The system shall convert the image to a 2d byte array
- R1.4. The system shall iterate over the 2d byte array

R2. The system shall accept and process a height field.

- R2.1. The system shall locate the height field file in a specific input folder.
- R2.2. The system shall read the height field file.
- R2.3. The system shall convert the height field to a 2d byte array.

R3. The system shall accept adjustable parameters such as baseline distance of the distorted image.

- R3.1. The system shall use the value of the baseline distance in the De Marzio function later in the program.

R4. The system shall validate that the image, height field, and parameters are acceptable

- R4.1 Verify the given ortho-rectified image is of the appropriate size and resolution.
- R4.2 Verify the given height field image is of the appropriate size and resolution.
- R4.3 Verify parameters are the correct length and type.

R5. The system shall match values from the height field to each pixel of the ortho-corrected image.

- R5.1. The system shall iterate through the ortho-corrected image and identify the pixel's color value at each location.
- R5.2. The system shall assign each pixel value a distortion value that is sourced from whatever value is in the height field table at the same location.

R6. The system shall create a new, distorted image.

- R6.1. The system shall create a function that uses the De Marzio equation to use the location of a pixel value and the distortion value associated with it to determine the new location of the pixel value in the distorted image.
- R6.2. The system shall use each pair of pixel values and distortion values as inputs in the De Marzio function.
- R6.3. The system shall use the new location of each pixel value to construct a new image.

R7. The system shall convert the distorted image and the ortho-corrected image into a height field to validate the accuracy of the distorted image

- R7.1 The system shall use an alternative version of the De Marzio equation using the original ortho-rectified and the new distorted image to make a new height field.
  - The de marzio equation is an equation that is used for calculating the shift from regular images to ortho-corrected images. More than likely this equation will be slightly edited, as we were given a confusing 20 year old chunk of code to decipher. Somewhere in it is the equation.
- R7.2 The new height field shall be compared to the original height field to ensure that they are mostly identical.
  - This will put the two images side by side to see if they do indeed look mostly identical

1. Performance Requirements
   - The system shall save this new image to an output folder.
   - Time-related performance requirements shall be proportional to the size of the images being adjusted, and the acceptable range of runtimes shall be made in collaboration with the customer.
   - The program shall be written in Python.

2. Design Constraints
   - The system will not create more than one image stereo-pair per run.
   - The system will not run on more than one computer at a time.
   - The system will not output the created image on the screen.

    - These constraints are present because we worry about the speed and efficiency of python. Unfortunately we do not want to write entire new libraries to make this a semi-fast program. It will just have to be run one at a time.

3. Quality Requirements

- The created image shall maintain the image clarity and color range from the original image and depth-field.
- The program shall be able to receive different-sized image inputs.
- The orthorectified image, depth field, and new distorted image files shall all be kept secure throughout the process through information hiding.
- The final, distorted image shall be within 95% identical to an actual distorted image taken from a satellite based on pixel-to-pixel comparison of the images.
- The final distorted image shall be combined with the depth field to create a new ortho-rectified image that shall be 95% identical to the original orthorectified image.
    - This all will be done using the equation provided in the folder of C files by the customer

Version: 1.0
6. December 2023

System Models
The System Models section will provide graphical system models that show the relationship between system components and the system environment. It will cover domain models, Use Case models, and Main Classes diagrams.

Lead Author: Matthew Isbell
Lead Author Signature:
[illegible]

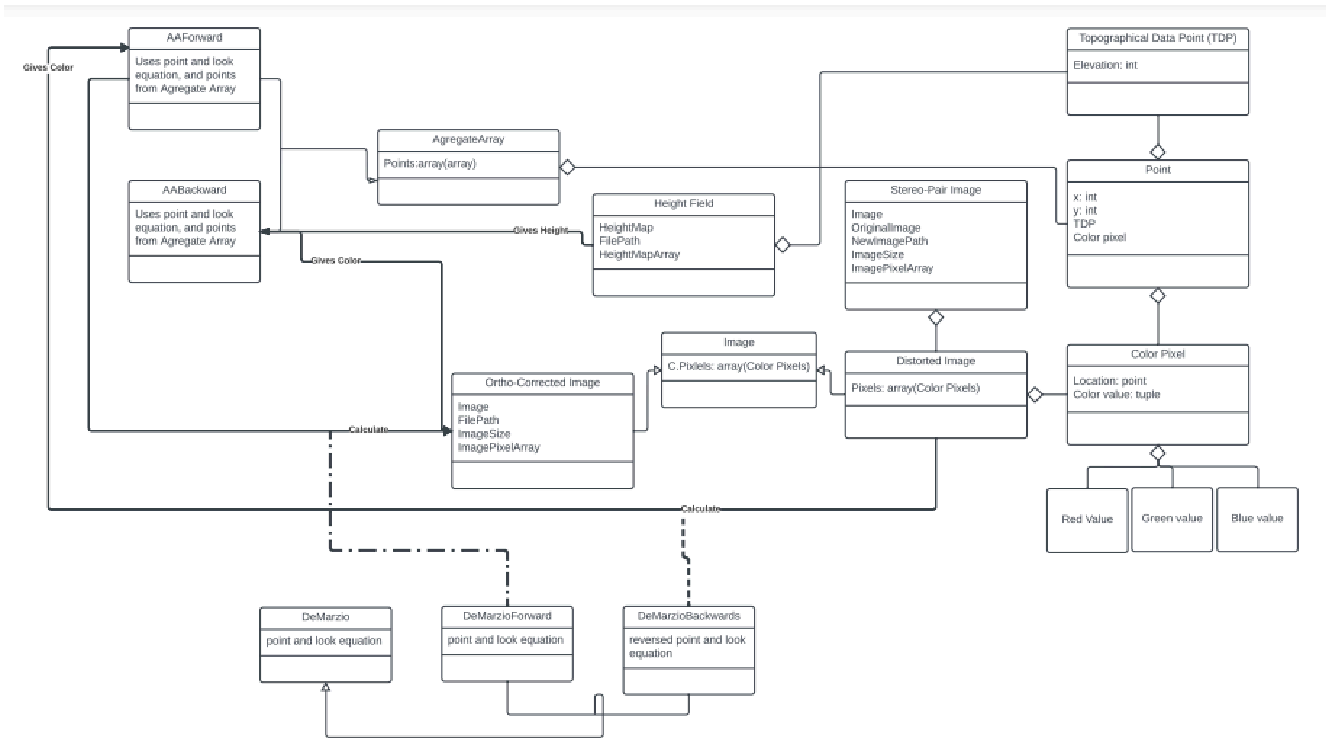
Review Names: Khaliu Olonbayar, Peter Kahnke, Scott Shlanta
Review Name Signatures:
[illegible]

[illegible]

[illegible]

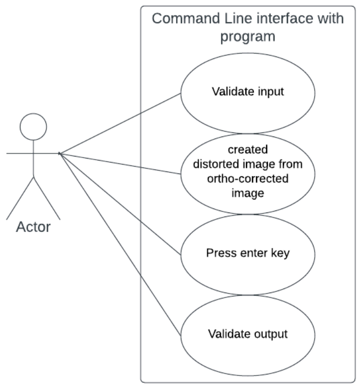
Augustana University
2001 S. Summit Ave
57197 Sioux Falls, SD

Domain Model



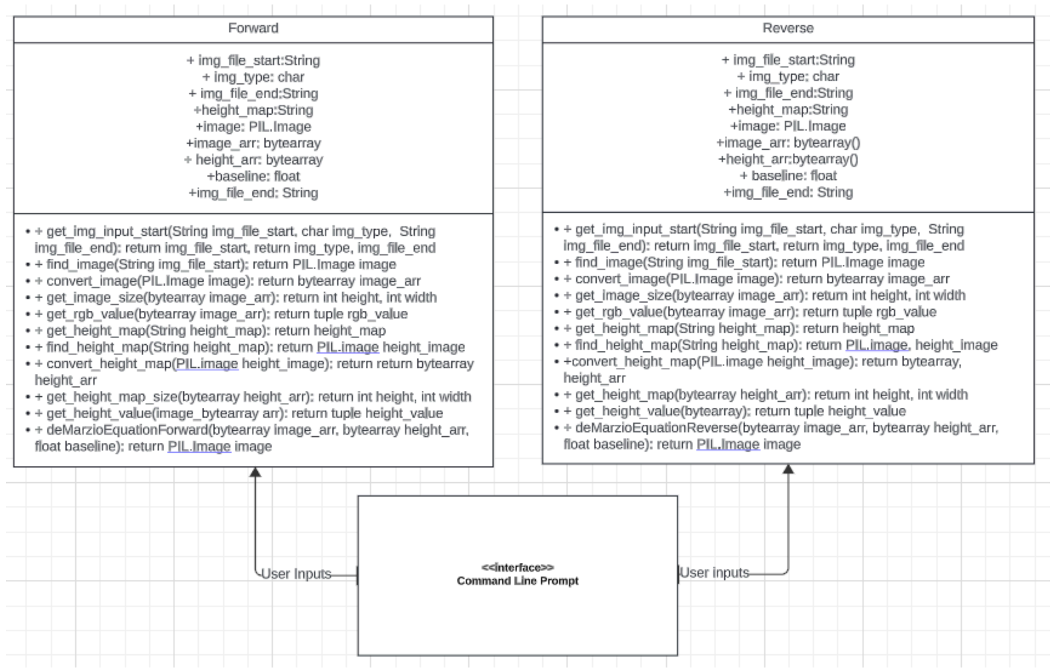
We will take in a distorted image provided by the customer and get the color values of each pixel. This is the point and color pixel section. It will then be put into an array of points, where it is put into a function containing the equation the customer provided. From here, we will also provide height map color values from 0 to 255. These are gathered in the TDP section, and also put into point. The function containing the De Marzio Equation will then create an original image that appears flat. There will also be a reverse function that takes in the flat image pixel values, the height map values, and then output an ortho-corrected image. We believe the equation name is De Marzio, so that is why we have three little De Marzio sections underneath the domain model.

Use Cases


Command Line interface with program
Validate input
created distorted image from ortho-corrected image
Press enter key
Validate output
Actor

the user will be able to interact with multiple sections of the program. The first being the input parameters. The user will input numerous fields (such as image name, height field name, output location), then press enter to begin the creation process. After that is complete, the user can then go to the desired output location and verify that this output was done correctly.

Main Classes



```
Forward

+ img_file_start:String
+ img_type: char
+ img_file_end:String
+height_map:String
+image: PIL.Image
+image_arr: bytearray
+ height_arr: bytearray
+baseline: float
+img_file_end: String

• + get_img_input_start(String img_file_start, char img_type,  String
  img_file_end): return img_file_start, return img_type, img_file_end
• + find_image(String img_file_start): return PIL.Image image
• + convert_image(PIL.Image image): return bytearray image_arr
• + get_image_size(bytearray image_arr): return int height, int width
• + get_rgb_value(bytearray image_arr): return tuple rgb_value
• + get_height_map(String height_map): return height_map
• + find_height_map(String height_map): return PIL.image height_image
• + convert_height_map(PIL.image height_image): return return bytearray
  height_arr
• + get_height_map_size(bytearray height_arr): return int height, int width
• + get_height_value(image_bytearray arr): return tuple height_value
• + deMarzioEquationForward(bytearray image_arr, bytearray height_arr,
  float baseline): return PIL.Image image
```

```
Reverse

+ img_file_start:String
+ img_type: char
+ img_file_end:String
+height_map:String
+image: PIL.Image
+image_arr: bytearray()
+height_arr:bytearray()
+ baseline: float
+img_file_end: String

• + get_img_input_start(String img_file_start, char img_type,  String
  img_file_end): return img_file_start, return img_type, img_file_end
• + find_image(String img_file_start): return PIL.Image image
• + convert_image(PIL.Image image): return bytearray image_arr
• + get_image_size(bytearray image_arr): return int height, int width
• + get_rgb_value(bytearray image_arr): return tuple rgb_value
• + get_height_map(String height_map): return height_map
• + find_height_map(String height_map): return PIL.image, height_image
• +convert_height_map(PIL.image height_image): return bytearray,
  height_arr
• + get_height_map(bytearray height_arr): return int height, int width
• + get_height_value(bytearray): return tuple height_value
• + deMarzioEquationReverse(bytearray image_arr, bytearray height_arr,
  float baseline): return PIL.Image image
```

We will have two classes. One class will create a flat image, and the other will create an ortho-corrected image. The class that creates a flat image will take in a file name, find the file on the computer, convert the image into a byte array, get the dimensions of the image, and the color values of each pixel. Then the user will enter the height map, get the dimensions of the image, return the height values, then put all of this information into the equation the customer provided to create a flat image. The process to create an ortho-corrected image is the exact same, except the last step is to put it into a modified version of the customers equation to create an ortho-corrected image.

Version: 1.0
13. December 2023

Test Plan and Risks
This section will go in depth into test plans, expected output, and potential risks to the program's integrity

Lead Author: Peter Kahnke
Lead Author Signature:

[illegible]

Review Names: Khaliu Olonbayar, Matthew Isbell, Scott Shlanta
Review Name Signatures:

[illegible]

[illegible]

[illegible]

Augustana University
2001 S. Summit Ave
57197 Sioux Falls, SD

Test cases

#	Description	Input	Expected Output
1	(R1.1) Tests for whether the program can locate an ortho-corrected image in the expected folder	Path to the folder containing the ortho-corrected image	The image is successfully found
2	(R1.2) Tests whether the program can read the ortho-corrected image	Ortho-corrected image	Image file can be read
3	(R1.3) Tests whether the system converts the image into a 2D byte array successfully	Ortho-corrected image	A 2D byte array is made of the image
4	(R1.4) Tests whether the program can iterate over the 2D byte array	2D byte array of image	Program fully iterates over array without errors
5	(R2.1) Tests whether the system can locate the height field in the expected folder	Path to folder containing the height field	The height field is successfully found
6	(R2.2) Tests whether the system reads the height field file correctly	Height field file	File can be read
7	(R2.3) Test whether the system converts the height field into a 2D byte array successfully	Height field file	A 2D byte array is made of the height field
8	(R3.1) Test whether the system successfully uses the De Marzio Equation	Baseline distance parameters	Confirmation that the the De Marzio Equation was used successfully

We have eight (8) test cases that we will verify.

1. This test will focus on the file system capabilities of the program. A user will enter a file path leading to the ortho-corrected image. The test will be if the program can successfully locate the image at that location. The risk in this test would be the user mistyping the path. If that were to happen, the program would either not find the image or locate an incorrect image.
2. This test case will work in conjunction with the previous test (test 1). Based on the path entered and the image at that location, this test will attempt to read in that file. A successful test would result in accessible data from the image. The risk of this test is the fact that we would be relying on an external file system. This means that we will have to work around different operating systems.
3. This test case will verify the algorithm that convert the ortho-corrected image to a 2d byte array. It will verify that the iterated over pixels are appropriately reflected in an appropriately sized 2d byte

array. The risk in this case is that the pixels have three values associated with them (red, green, and blue). We will need to be sure that these three values are reflected in each index of the 2d array.

4. This test will test to be sure that the 2d array made in the previous step is accessible. It will confirm that a loop can access each element. A successful output would be one devoid of errors. The risk here is that it is reliant on the previous step going to plan.
5. This test, similar to step one, will focus on the file system capabilities of the program. A user will enter a file path leading to the height field. The test will be if the program can successfully locate the height field at the given location. The risk in this test would be the user mistyping the path. If that were to happen, the program would either not find the height field or locate an incorrect image.
6. This test will focus on the programs ability to read in the height field image. Based on the path entered in the previous step, the image would be found and appropriately read in without errors. The risk is the difference between operating systems used to run the program.
7. This test would confirm that the program can covert the height field into a 2d byte array containing height values. The risk is that height fields are generally different file types that the images. Additionally, they are black and white, meaning that we need to decode the values to an extent.
8. This test will validate the effectiveness of the De Marzio equation. The test will focus on the created distorted image vs an original distorted image and how they compare. Part of this test is running the whole process in reverse (i.e. going from a flat image to ortho-recitified). This reverse process will give up two copies of each image to test against each other. The risk here is that for this step to work, every other step must be performed correctly.

Version: 1.0
13. December 2023

Project Evolution
This section with go over project assumptions, potential changes to needs, and potential hardware changes

Lead Author: Peter Kahnke
Lead Author Signature:

[illegible]

Review Names: Khaliu Olonbayar, Matthew Isbell, Scott Shlanta
Review Name Signatures:

[illegible]

SShlanta

[illegible]

Augustana University
2001 S. Summit Ave
57197 Sioux Falls, SD

The assumptions of this project:

1. The users of this project has access to the appropriate file types and data found on their computer
2. The selected image and height field corresponds to the same field of view taken by the camera
3. The users computer is capable of executing the program
4. This program is completed and has been tested to ensure proper functionality

Potential changes in user needs:

1. Users may require an adjustment in the program to meet new file types
2. The user may want to change the hardware that the program runs on. This will not be of particular concern because the program is entirely code and is run on the command line.

Future changes:

1. As previously mentioned, we may need to adjust the program to meet new file types
2. We may implement a GUI at a future point to increase user understanding
3. We may implement a way for multiple images to be processed at once

Version: 1.0
14. December 2023

Appendices
This section will give a recommend software layout in a simple, text-based way.

Lead Author: Peter Kahnke
Lead Author Signature:

[illegible]

Review Names: Khaliu Olonbayar, Matthew Isbell, Scott Shlanta
Review Name Signatures:

[illegible]

[illegible]

[illegible]

Augustana University
2001 S. Summit Ave
57197 Sioux Falls, SD

Recomended software layout:

1. (Class) Forward
    a. This class will run a conversion from ortho-corrected image to flat image
        i. We are programming this in python due to extensive libraries and image processing capabilities
        ii. The only hardware component here is the computer that the program runs on
        iii. To execute the task at hand, we will do the following:
            1. Read in the starting image (ortho-corrected) based on an inputted file path
            2. Convert the ortho-corrected image to a byte array of tuples containing 3 RGB (Red, Green, and Blue) values
            3. Read in the height field based on an inputted file path
            4. Convert the height field to a byte array of height values
            5. Using these two byte arrays use the De Marzio equation to reassign pixels to their originally appropriate location.
2. (Class) Reverse
    a. This class will run a conversion from flat image to ortho-corrected image to use as a testing standard for Class Forward
        i. We are programming this in python due to extensive libraries and image processing capabilities.
        ii. The only hardware component here is the computer that the program runs on
        iii. To execute the task at hand, we will do the following:
            1. Read in the starting image (flat) based on an inputted file path
            2. Convert the ortho-corrected image to a byte array of tuples containing 3 RGB (Red, Green, and Blue) values
            3. Read in the height field based on an inputted file path
            4. Convert the height field to a byte array of height values
            5. Using these two byte arrays use the De Marzio equation to reassign pixels to their ortho-corrected location.

Version: 1.0
13. December 2023

Index

Lead Author: Khaliu Olonbayar
Lead Author Signature:

[illegible]

Review Names: Khaliu Olonbayar, Matthew Isbell, Scott Shlanta
Review Name Signatures:

[illegible]

[illegible]

[illegible]

Augustana University
2001 S. Summit Ave
57197 Sioux Falls, SD