Stereo Pair Retrogenerator

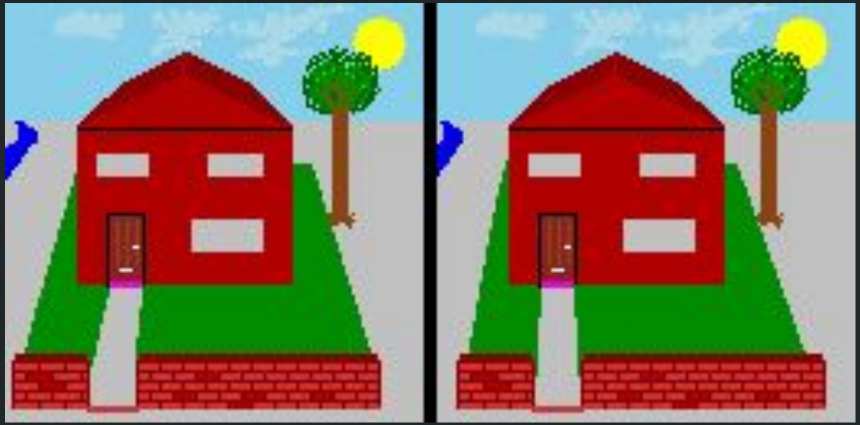
The For (Four) Loops

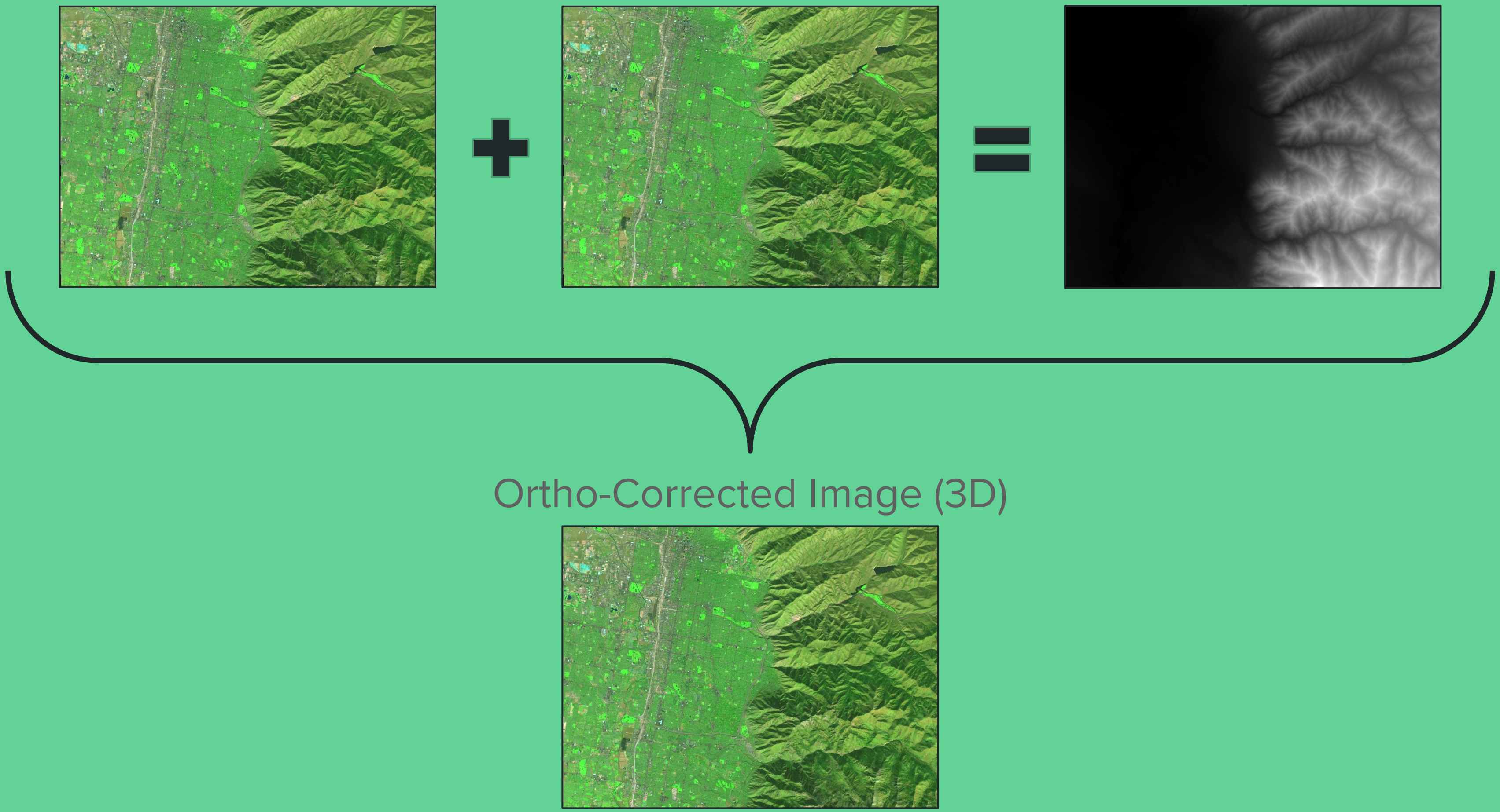
Scott Shlanta, Peter Kahnke, Matthew Isbell, Khaliu Olonbayar

Background Info

- Stereo-pair: two slightly different images that look 3D when combined

- Human brains do this automatically, but computers do not.





Left-Eye View (Flat) + Right-Eye View (Flat) = Height Field

Ortho-Corrected Image (3D)

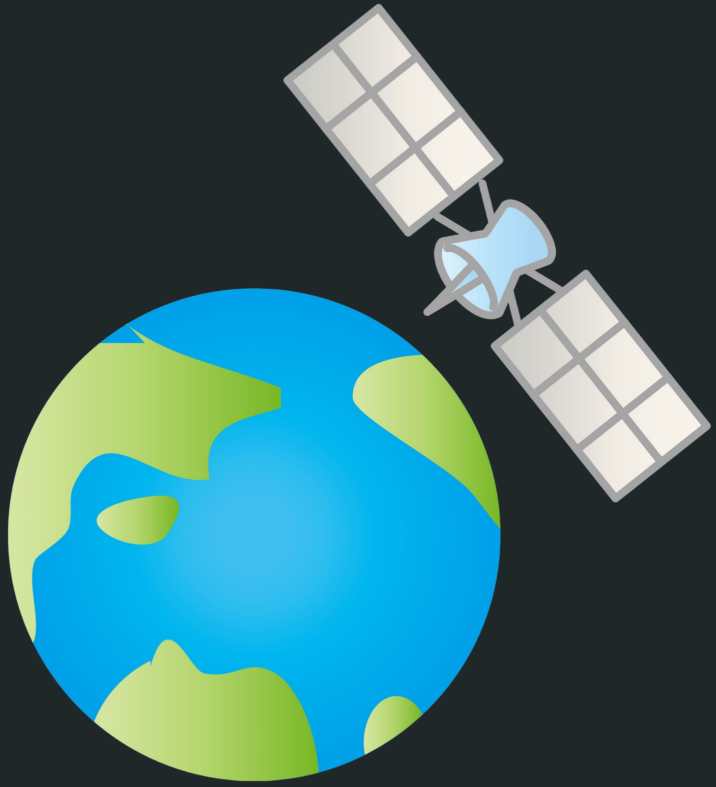
The Twist ?

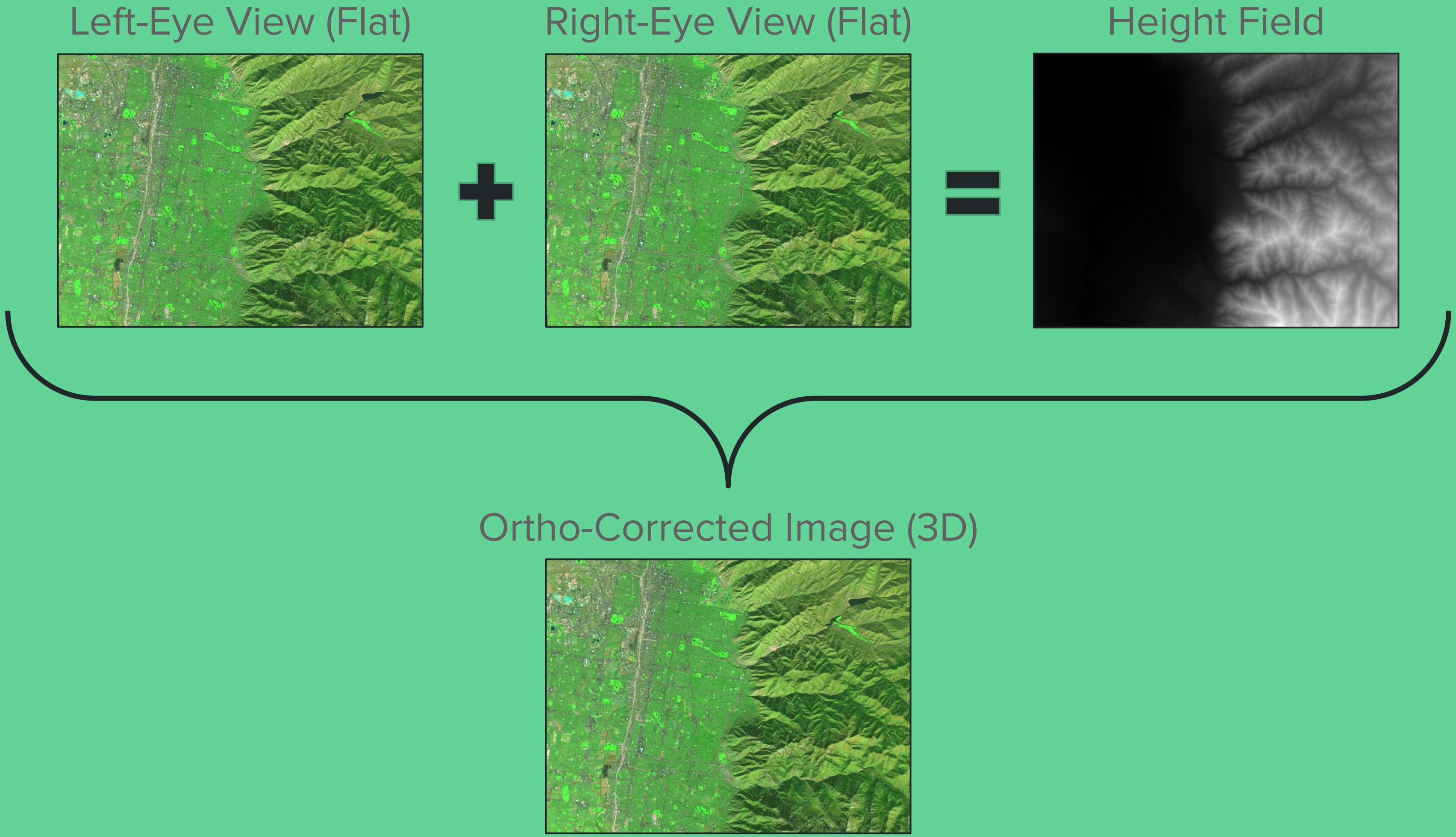
Do it all backwards...

What a twist!

Project Overview

- Can you make a flat image from an ortho-corrected image and a height field?
- Can you adjust distortion according to baseline distance.
- The users for this project will be experts in satellite imagery and work regularly in the field

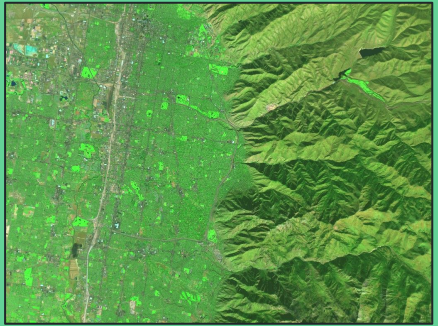




Left-Eye View (Flat) + Right-Eye View (Flat) = Height Field

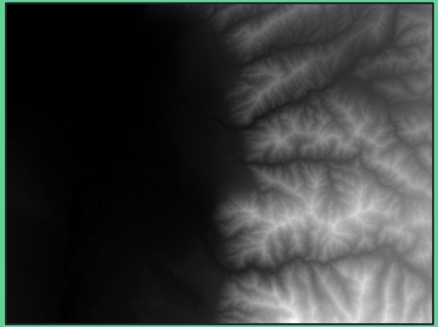
Ortho-Corrected Image (3D)

Ortho-Corrected Image (3D)



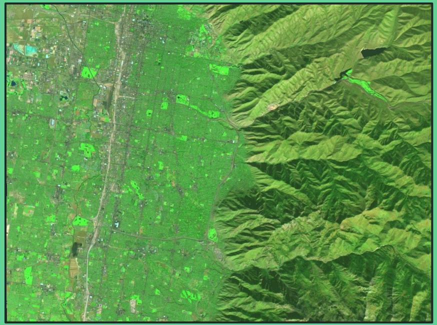
+

Height Field



=

Left-Eye View (Flat)



Design Requirements for Program

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.
- R7.2 The new height field shall be compared to the original height field to ensure that they are mostly identical.

Design Requirements for Program

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.

R7.2 The new height field shall be compared to the original height field to ensure that they are mostly identical.

Misc. Requirements for Program

1.1 Performance Requirements

1.1.1. 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.

1.1.2. The program shall be written in Python.

1.2. Design Constraints

1.2.1. The system will not create more than one image stereo-pair per run.

1.2.2. The system will not run on more than one computer at a time.

1.2.3. The system will not output the created image on the screen.

1.3. Quality Requirements

1.3.1. The created image shall maintain the image clarity and color range from the original image and depth-field.

1.3.2. The program shall be able to receive different-sized image inputs.

1.3.3. The orthorectified image, depth field, and new distorted image files shall all be kept secure throughout the process through information hiding.

1.3.4. 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.

1.3.5. 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.

Misc. Requirements for Program

1.1 Performance Requirements

1.1.1. 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.

1.1.2. The program shall be written in Python.

1.2. Design Constraints

1.2.1. The system will not create more than one image stereo-pair per run.

1.2.2. The system will not run on more than one computer at a time.

1.2.3. The system will not output the created image on the screen.

1.3. Quality Requirements

1.3.1. The created image shall maintain the image clarity and color range from the original image and depth-field.

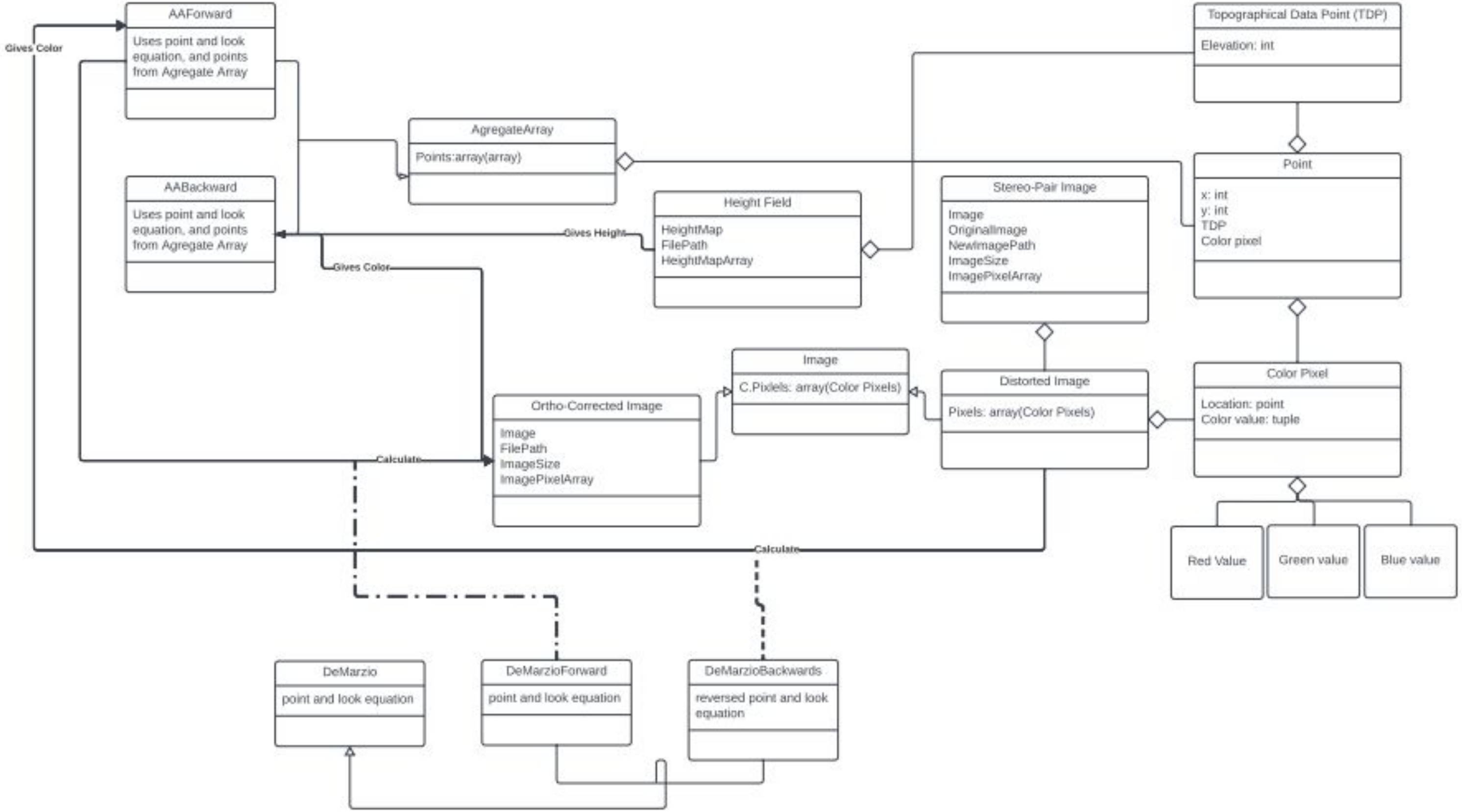
1.3.2. The program shall be able to receive different-sized image inputs.

1.3.3. The orthorectified image, depth field, and new distorted image files shall all be kept secure throughout the process through information hiding.

1.3.4. 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.

1.3.5. 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.

Domain Model



AAForward

Uses point and look equation, and points from Agregate Array

AABackward

Uses point and look equation, and points from Agregate Array

AgregateArray

Points:array(array)

Height Field

HeightMap
FilePath
HeightMapArray

Topographical Data Point (TDP)

Elevation: int

Point

x: int
y: int
TDP
Color pixel

Stereo-Pair Image

Image
OriginalImage
NewImagePath
ImageSize
ImagePixelArray

Image

C.Pixels: array(Color Pixels)

Distorted Image

Pixels: array(Color Pixels)

Color Pixel

Location: point
Color value: tuple

Ortho-Corrected Image

Image
FilePath
ImageSize
ImagePixelArray

Red Value

Green value

Blue value

DeMarzio

point and look equation

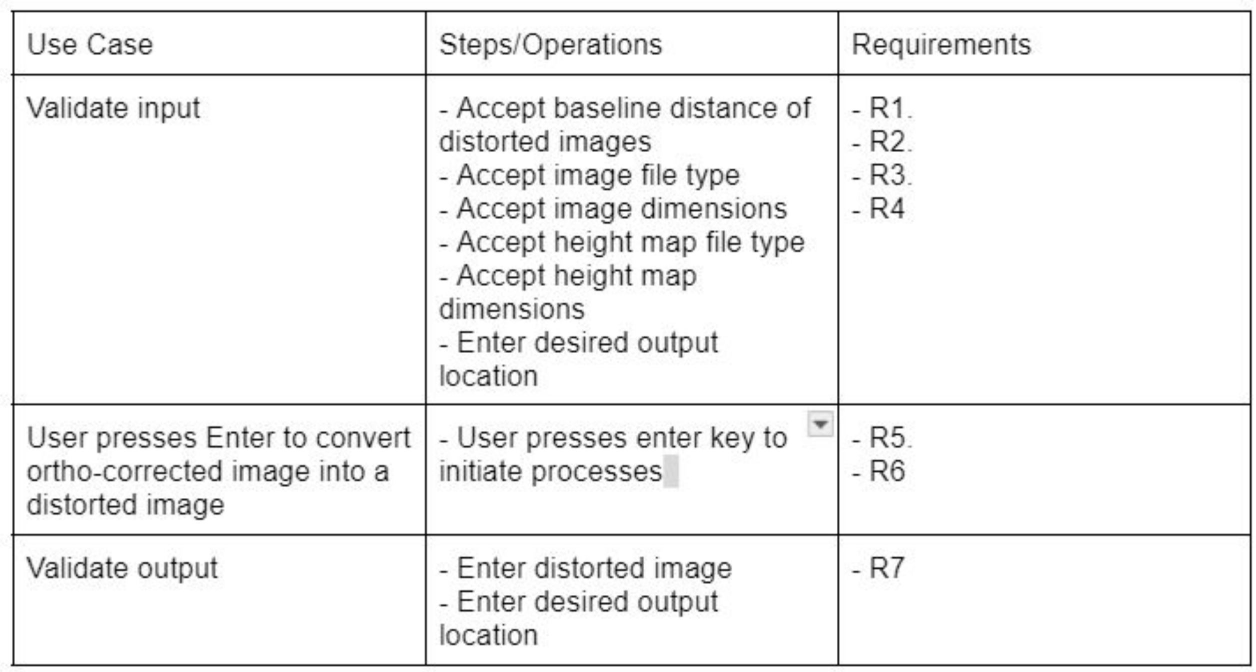
DeMarzioForward

point and look equation

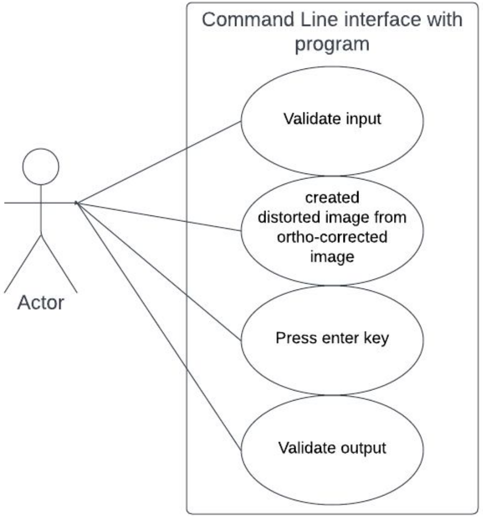
DeMarzioBackwards

reversed point and look equation

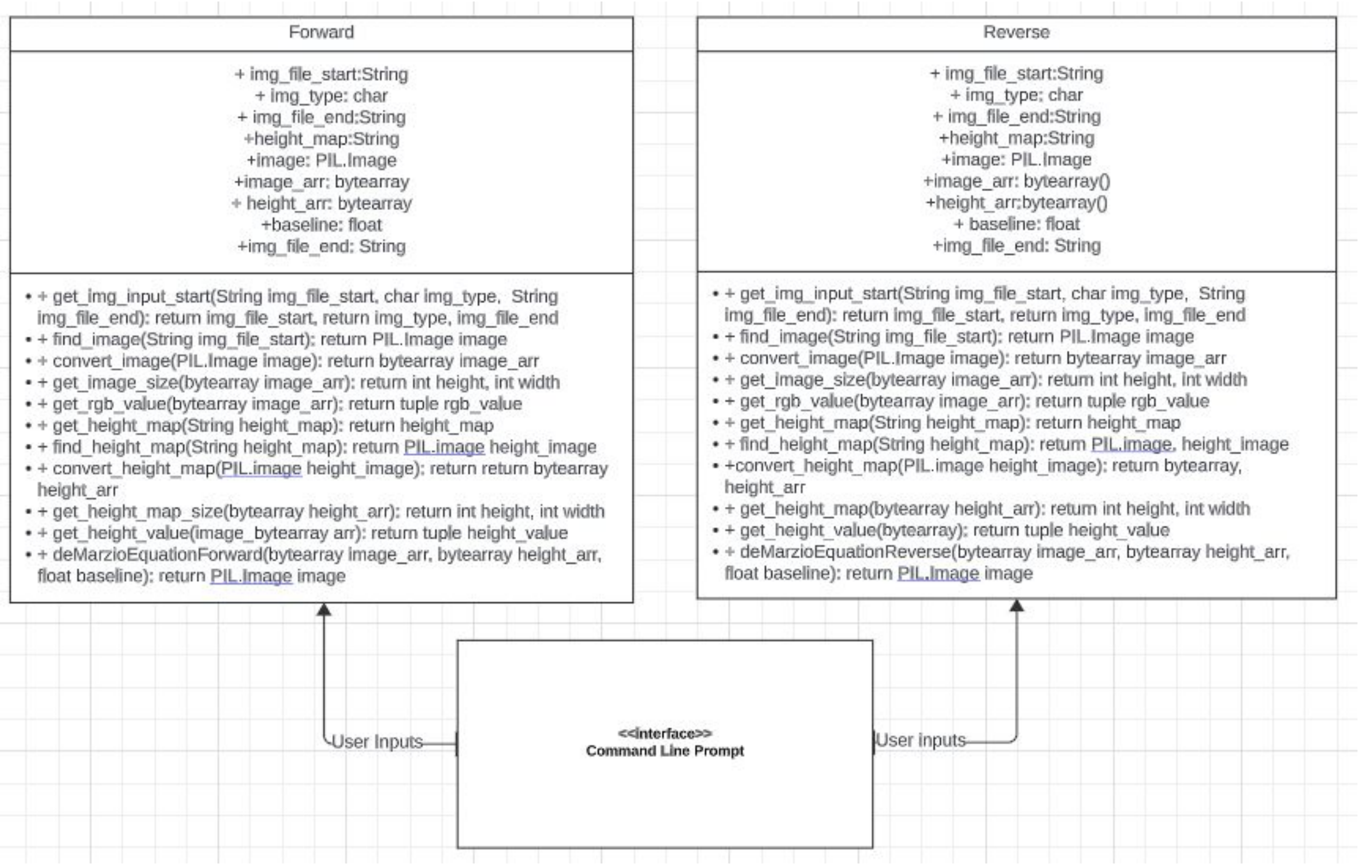
Use Cases



Use Case	Steps/Operations	Requirements
Validate input	- Accept baseline distance of distorted images - Accept image file type - Accept image dimensions - Accept height map file type - Accept height map dimensions - Enter desired output location	- R1. - R2. - R3. - R4
User presses Enter to convert ortho-corrected image into a distorted image	- User presses enter key to initiate processes	- R5. - R6
Validate output	- Enter distorted image - Enter desired output location	- R7


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

Main classes for Implementation



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

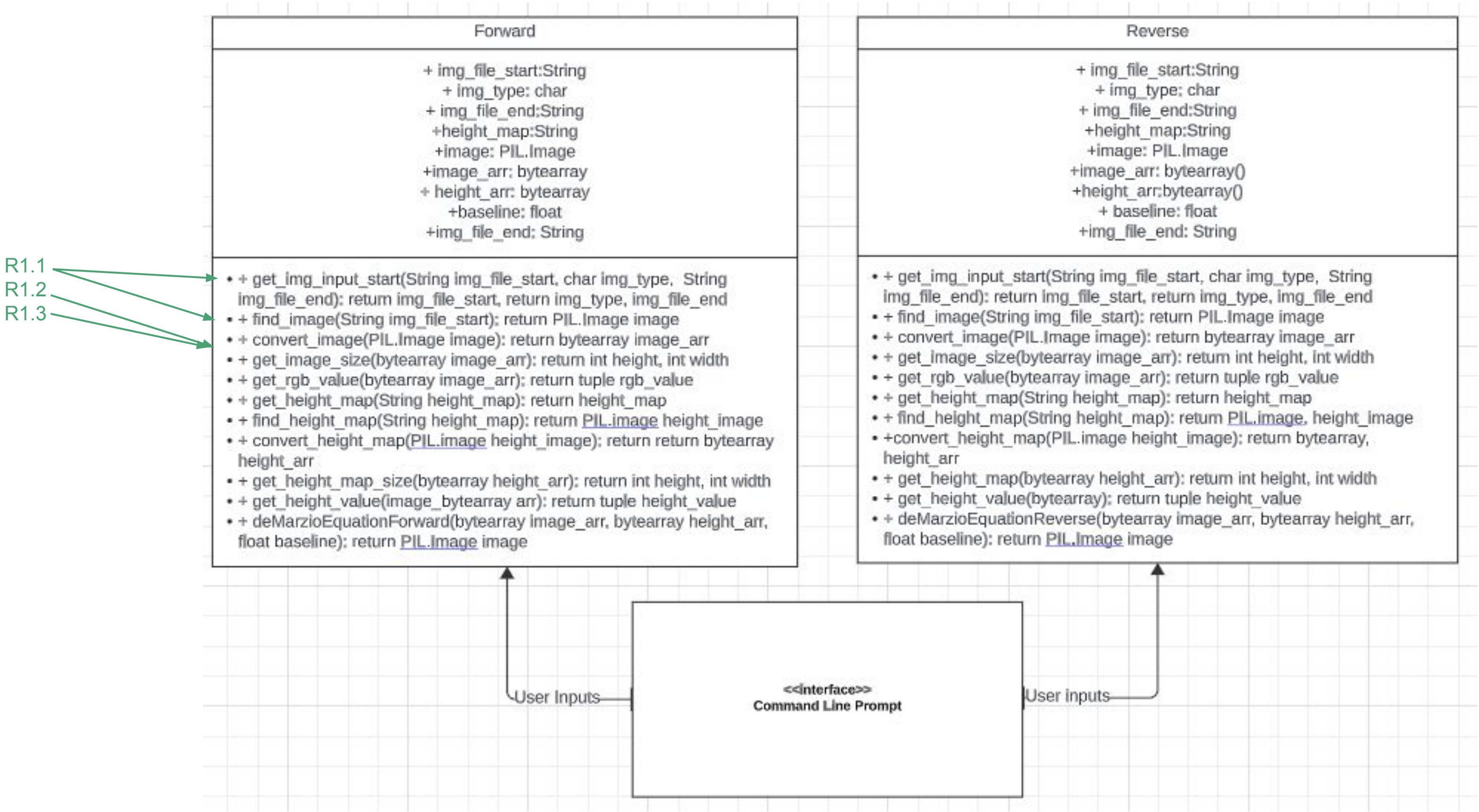
Reverse

+ 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

<<interface>>
Command Line Prompt

Main classes for Implementation



R1.1

R1.2

R1.3

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

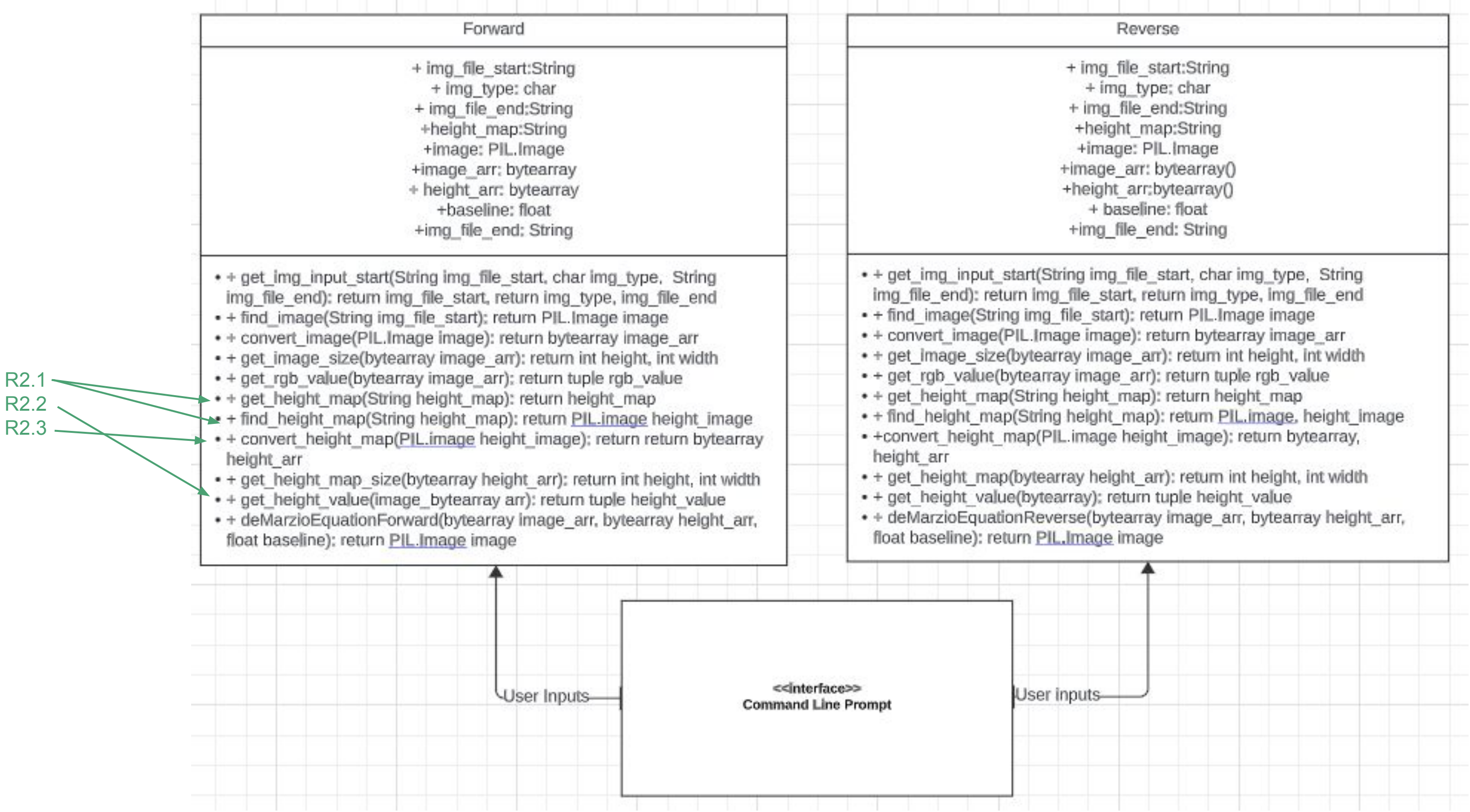
- + get_img_input_start(String img_file_start, char img_type, String img_file_end): return img_file_start, return img_type, img_file_end
- + find_image(String img_file_start): return PIL.Image image
- + convert_image(PIL.Image image): return bytearray image_arr
- + get_image_size(bytearray image_arr): 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

Main classes for Implementation



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

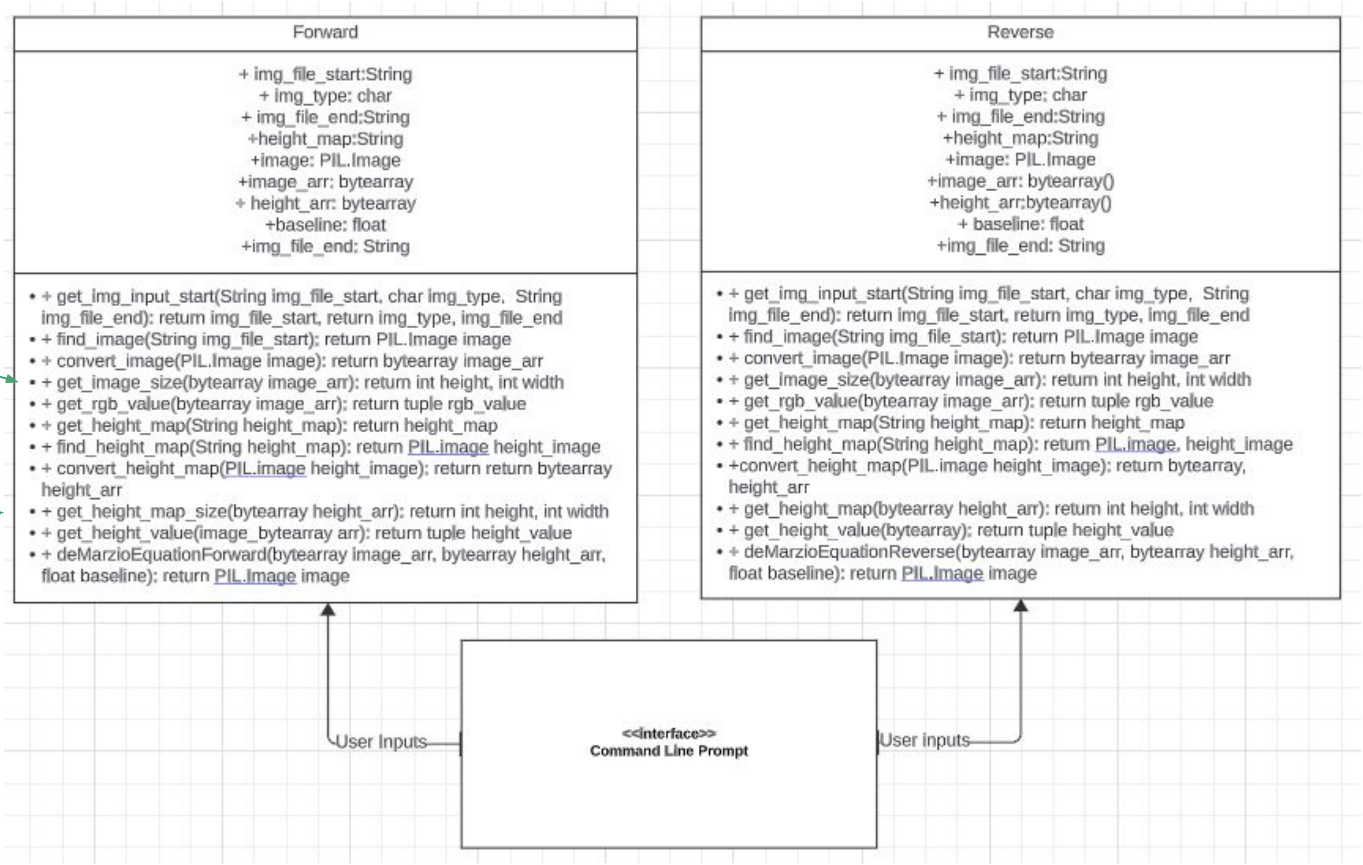
Reverse

+ 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

<<interface>>
Command Line Prompt

Main classes for Implementation



R4.1

R4.2

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

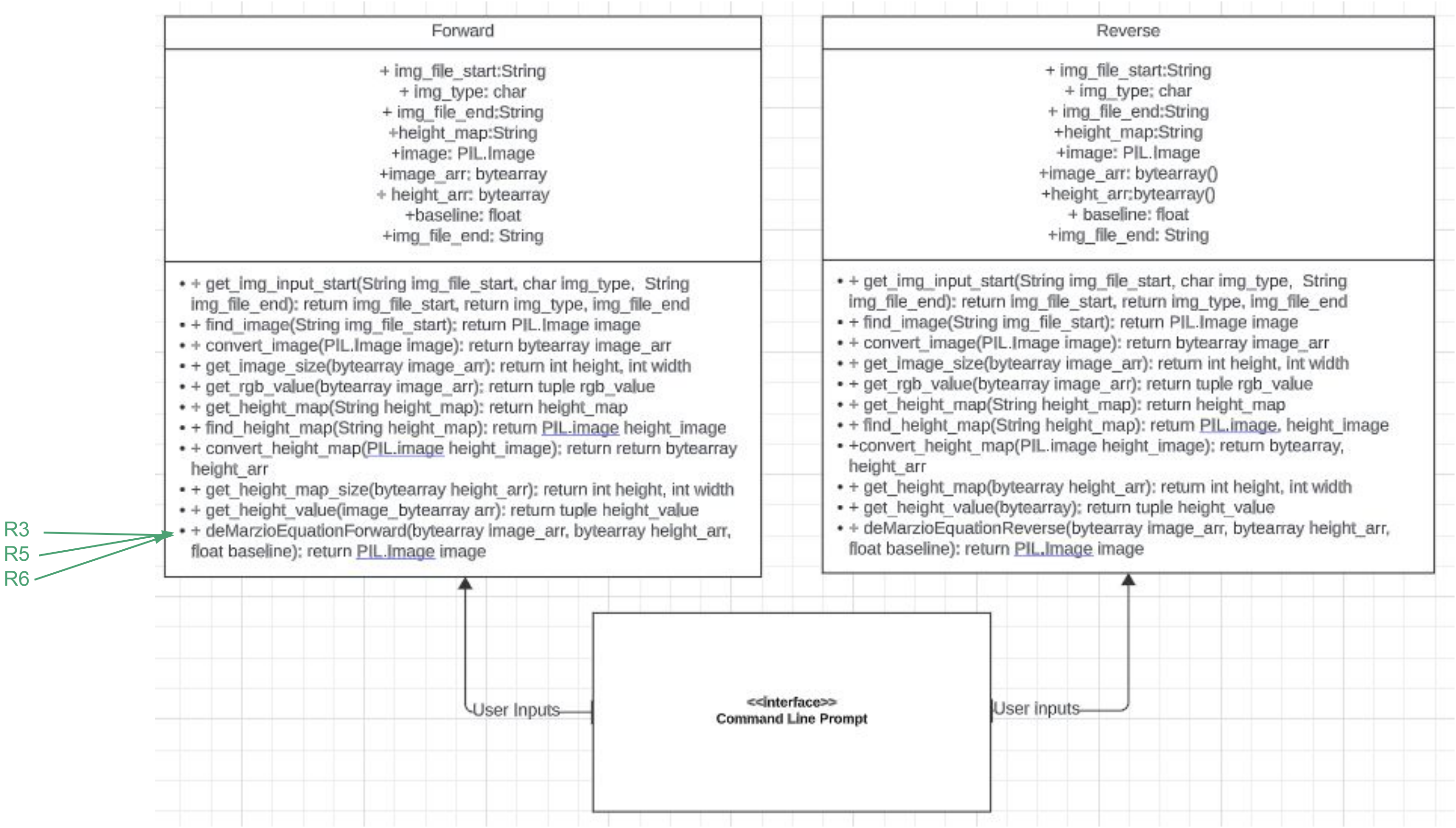
Reverse

+ 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

<<interface>>
Command Line Prompt

Main classes for Implementation



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

R3
R5
R6

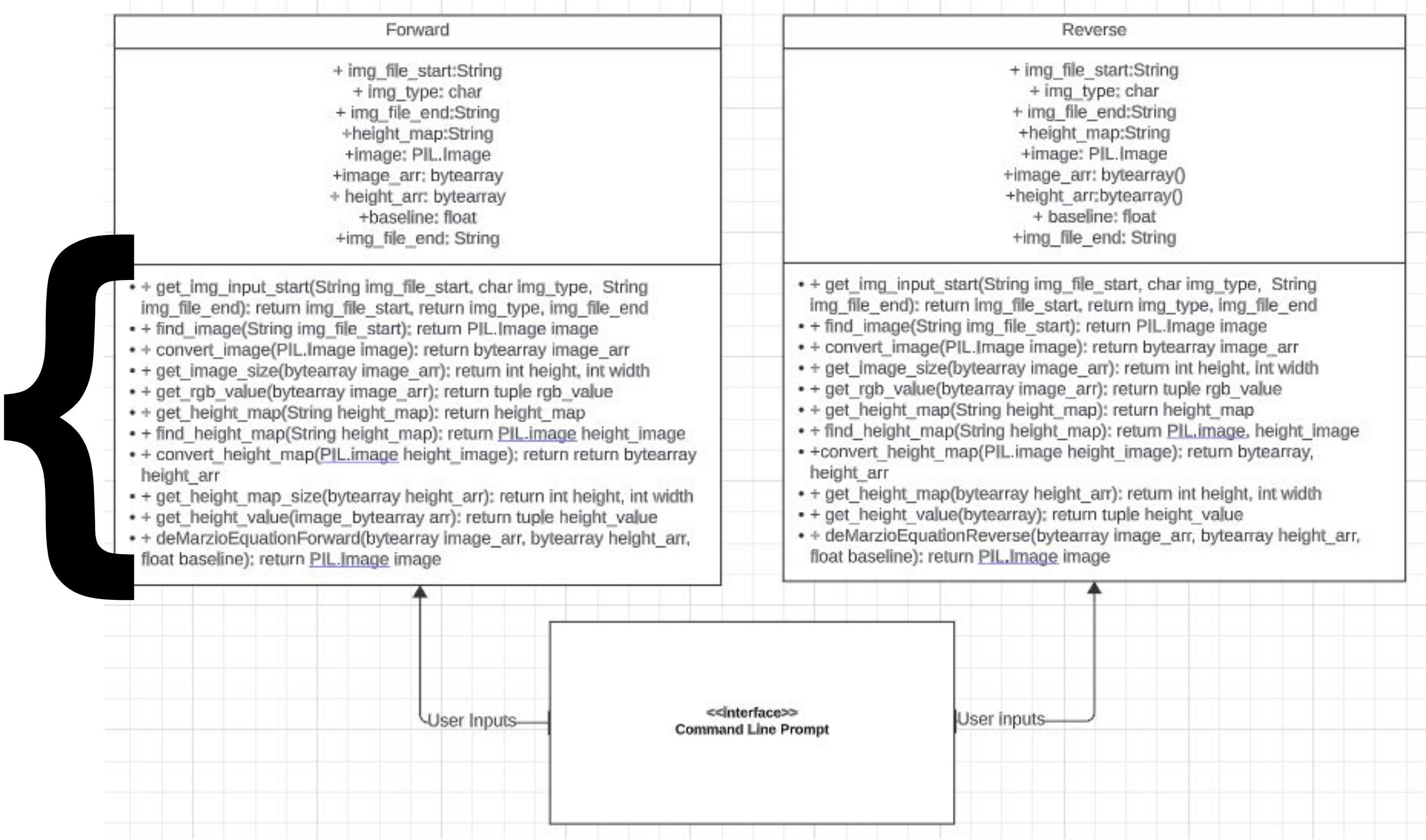
Reverse

+ 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

Main classes for Implementation

R7



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

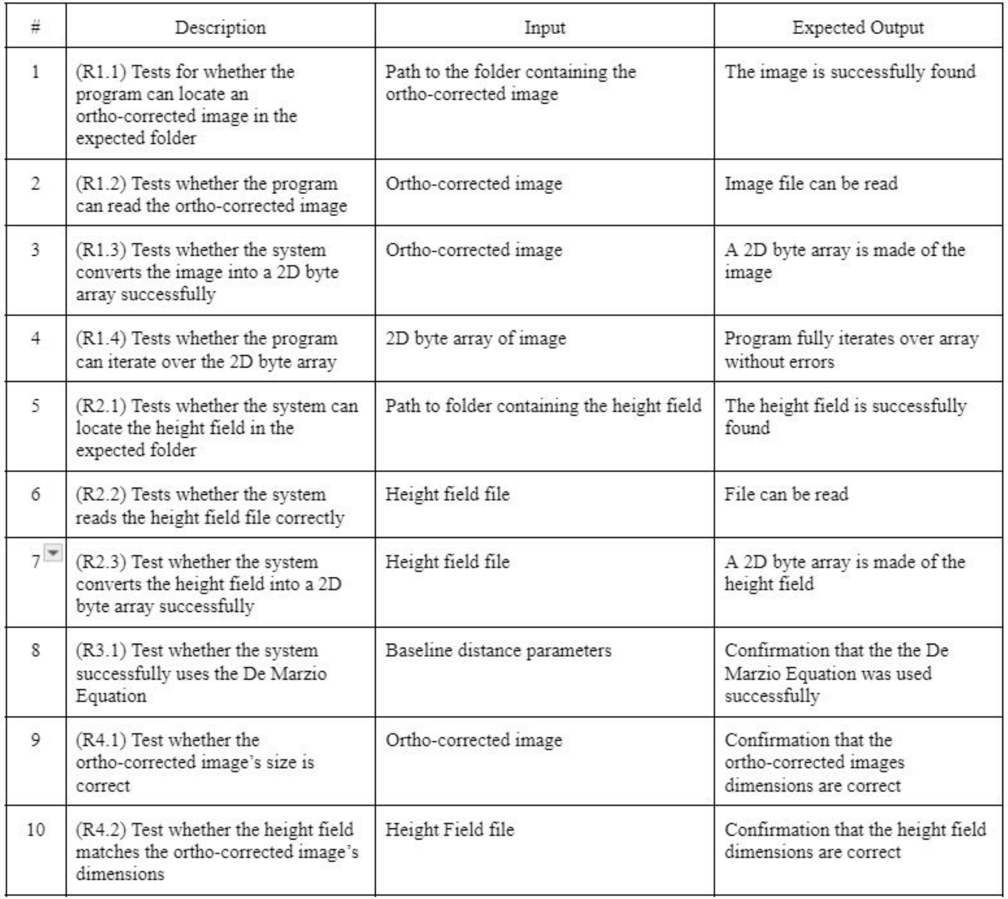
- + get_img_input_start(String img_file_start, char img_type, String img_file_end): return img_file_start, return img_type, img_file_end
- + find_image(String img_file_start): return PIL.Image image
- + convert_image(PIL.Image image): return bytearray image_arr
- + get_image_size(bytearray image_arr): 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

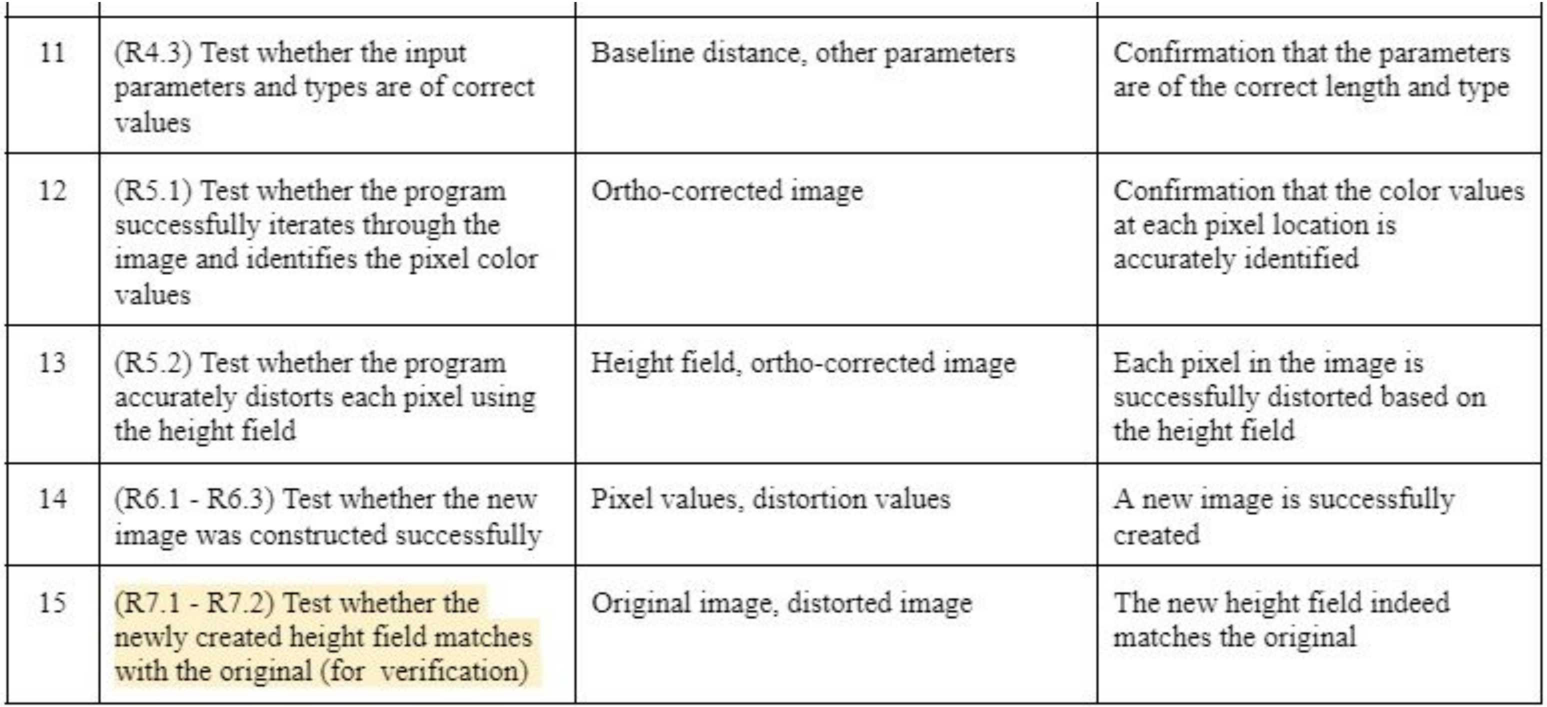
- + get_img_input_start(String img_file_start, char img_type, String img_file_end): return img_file_start, return img_type, img_file_end
- + find_image(String img_file_start): return PIL.Image image
- + convert_image(PIL.Image image): return bytearray image_arr
- + get_image_size(bytearray image_arr): 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

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
9	(R4.1) Test whether the ortho-corrected image's size is correct	Ortho-corrected image	Confirmation that the ortho-corrected images dimensions are correct
10	(R4.2) Test whether the height field matches the ortho-corrected image's dimensions	Height Field file	Confirmation that the height field dimensions are correct

Test Cases cont.

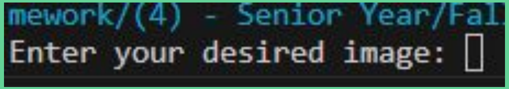


11	(R4.3) Test whether the input parameters and types are of correct values	Baseline distance, other parameters	Confirmation that the parameters are of the correct length and type
12	(R5.1) Test whether the program successfully iterates through the image and identifies the pixel color values	Ortho-corrected image	Confirmation that the color values at each pixel location is accurately identified
13	(R5.2) Test whether the program accurately distorts each pixel using the height field	Height field, ortho-corrected image	Each pixel in the image is successfully distorted based on the height field
14	(R6.1 - R6.3) Test whether the new image was constructed successfully	Pixel values, distortion values	A new image is successfully created
15	(R7.1 - R7.2) Test whether the newly created height field matches with the original (for verification)	Original image, distorted image	The new height field indeed matches the original

Before we head to the code...

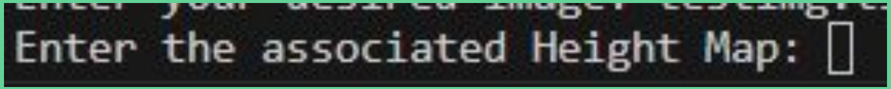
User Input

- Command Prompt only...
- Will ask the user for an image



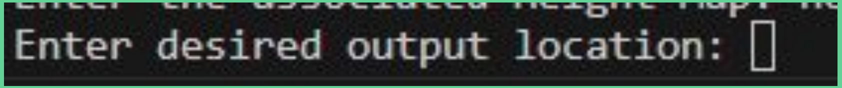
```
mework/(4) - Senior Year/Fal
Enter your desired image: 
```

- Will then ask for a height map associated with the image



```
Enter the associated Height Map: 
```

- Will ask for image output location

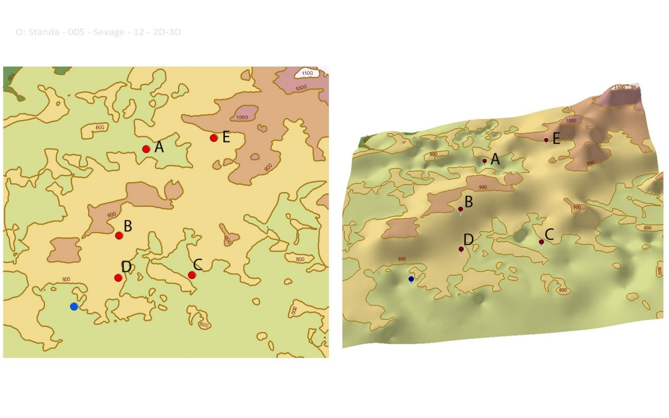


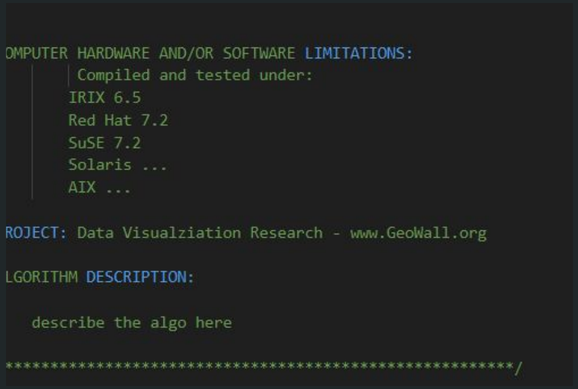
```
Enter desired output location: 
```

Enters Code

Future Maintenance

- How long will this take to implement?
- How do we turn 2D into 3D?
- How do we create the equation?
- How will we adapt to changing image types?





```
OMPUTER HARDWARE AND/OR SOFTWARE LIMITATIONS:
        Compiled and tested under:
       IRIX 6.5
       Red Hat 7.2
       SuSE 7.2
       Solaris ...
       AIX ...

ROJECT: Data Visualziation Research - www.GeoWall.org

LGORITHM DESCRIPTION:

   describe the algo here

********************************************************/
```

Questions?

End