**Image Processing and the STM**

PHYS 350 - Computational Physics

Paige Meyer

**Acknowledgment**

This problem came from Newman’s Computational Physics Book problem 23 in chapter 5. In addition, *Figure 1* and formulae also came from the statement of the problem.

**Introduction**

Many times in television shows, movies, and games the landscape is simulated through computer graphics. These graphics allow for a different, sometimes fictional setting. While other times, it may be out of the budget to travel to another place, and instead the background is simulated on the computer. The landscape could have mountain, plateaus, valleys, etc. One of the main hurdles in creating an environment is simulating light correctly. In this project, simulating light on terrain will be completed.

Overhead light tends to give every area on a flat surface an equal amount of light. When the light is at an angle to the surface, the brightness of the area seems to dim. Given the amount of light provided is constant, there is a relationship between the angle of the light source and observed intensity of the light at that point. If the light is perpendicular (normal) to a flat area with unit one, the light would “see” an area of one. Now, when the light is at an angle to the normal, the same amount of light hits a larger area. This makes the illumination of the surface decrease. As the angle increases, the illumination of the surface decreases. Therefore, the illumination of the surface is proportional to This phenomenon is depicted in *Figure 1*.

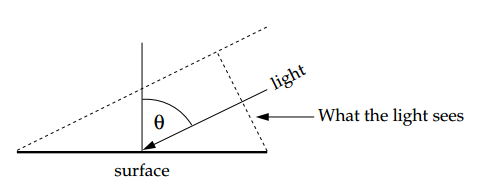


Figure 1 - Light sees less area as the angle increases [1].

Additionally, the brighter the light source is, the brighter the surface will be. This is saying the light intensity is proportional to the intensity of the light source Using this and the angle to the surface, we now find the illumination intensity is

This essentially means that the only component that gives brightness is the light’s y-component. This formula will allow computers to simulate the amount of light falling on land.

In the next section, we will discuss how we can solve using the height of various points of the objects, the intensity of the light, and the angle of the light to the surface. This will employ the central difference technique to find the slope of land. From this, the intensity of the light and the angle of the light source will be given. Therefore, we can solve for the illumination at each point. Firstly, it will be discussed how the slope was found.

**Methods**

Instead of changing the position of the light source, each point of land had a different slope. This would allow for the land to have hills, valley, and mountains. This means that for different areas, the normal will change, therefore the observed illumination will also change. This allows computer graphics to simulate hills, valleys and other mountainous regions illumination, allowing the land to be more realistic with the following conclusions.

Suppose, one wants to simulate the some area of land in the computer. The altitude of the each point is known. Earth’s surface equation is or rewriting This means that the normal vector to the surface is the gradient of . Therefore

Where a numerical approximation of the slope for the x-plane and y-plane using the central difference technique. Now, we have something that denotes if light was overhead. From before, we stated that we need to calculate with respect to the normal vector. We can do this by taking the dot product of the vector and. This is given by

where denotes the magnitude of each vector and is the angle between the light and the normal vector.

Rearranging, we have

Using the three previous equations, we find the surface’s illumination to be

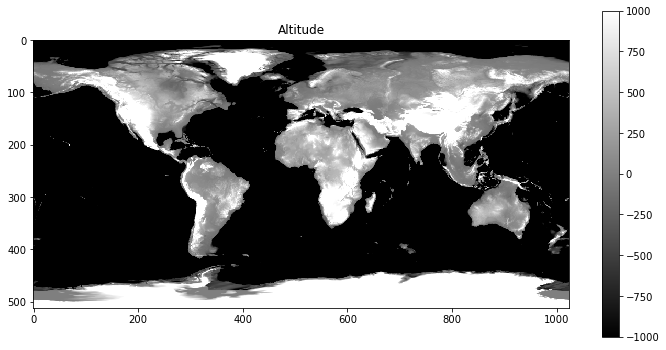
This equation allows use to find the illumination intensity at any point if we know the intensity of the light and land altitude. This would allow for real-world simulations of landscape.

This project only considers the case where light is horizontally shining with intensity one, such that where is the angle from horizontal. We know that the intensity illumination would simplify to

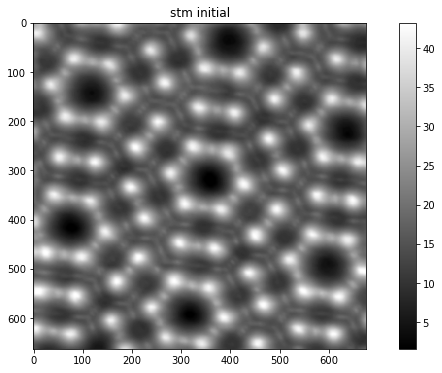
Speaking from a computational perspective, we found the slope in the x-direction and the slope in the y-direction using the central difference technique from the data of each point’s individual height. Then, the intensity of the illumination would be set by the programmer. This computation will be tested on data of a world map with altitudes taken 30,000 feet apart found in altitude.txt and a scanning tunneling microscope (STM) data, with heights measured 2.5 units apart found in stm.txt.

**Results**

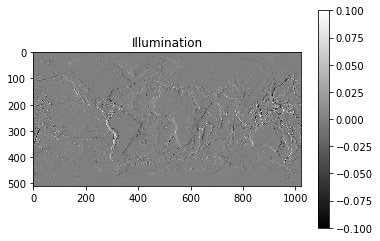
This was done with two data sets. The first of which is a world map. The altitudinal data was used. The altitude is shown in the figure below. The whiter the area, the higher it is. From here, it the Rocky, Himalayan, Alps, and Andes Mountain ranges can be seen. These will be the focus as the slopes here are changing steeply.



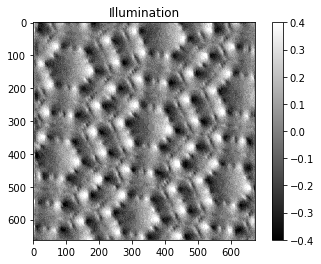
The second data set was from a STM, pictured below. Notice here, there are what look to be half spheres with the centers brighter than the rest. These will be the focus when trying to tell where the light is coming from.



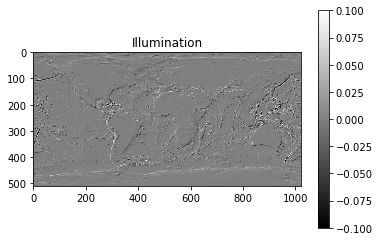
Using the case where the light is shining horizontally with intensity one, stated in the **Methods** section. This means that is simulates a light coming from one side of the image. This should have one side of a a mountain range very bright while the other side is in shadow. Using the **angle**  the light source should be seem as if it is to the left of the image. We find the altitude data agrees with the expectation by showing the mountainous areas being white on the side that is facing the light on the left. Meanwhile, the mountainous side tilted away from the light seems to be in the darker shadow.



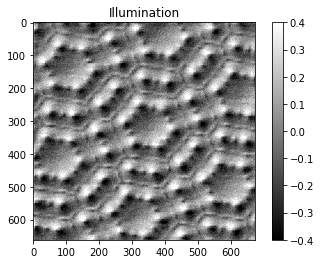
While the STM data with the same angle shows a similar result having the areas tilted toward the light lighter than the areas tilted away from the light. This is displayed in the image below.



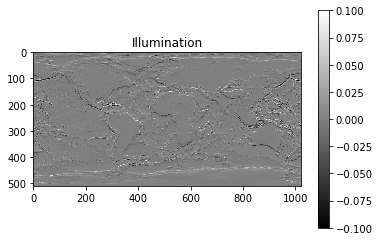
Now, when the **angle**, the bright parts of the higher areas should have rotated 45 degrees clockwise. In addition, the shadowed areas should have rotated 45 degrees clockwise as well. In other words, it should seem as if the light source is to the top left of the image. This is the case as we find when the altitude data shows the lighter side to be include the top some of the north-facing mountains. While the darker side to encompass more of the mountain sides facing the bottom right corner. This is shown in the image below.



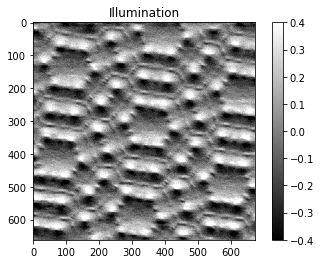
While the STM data shows a similar result with the same angle . The bright and dark areas seem to have rotated 45 degrees clockwise as shown in the following image.



Finally, it was tested where the **angle**, the bright part should be on the mountains’ north face, while the shadowed part would be on the south face since. Again, we verify this is the case with altitude data shows the mountain ranges north side being brighter than the south side as displayed in the image below.



While the STM data shows a similar result again with angle . The bright and dark areas seem to have rotated 45 degrees clockwise as shown in the following image. Where the light should seem as if it is coming from the top of the image.



All of these images agree with our intuition about where the light source should seem to be. The bright white parts are shown in white and the land tilted away from the light source seem to be in shadow. This seems to verify the program is working as expected.

**Conclusions**

Through this project, I learned why shining a flashlight horizontally is dimmer than shining a flashlight perpendicular to an object.

While developing this code, I tried using a numpy method for finding the dot product between the slope vector and the light intensity vector. Then, while testing I found I my variable for the angle did not affect the light source at all. After sitting on my code for a day, I came to the realization that I wasn’t using the angle at all because I was using the dot product, which didn’t need the angle between the vectors. I devised a way to change this by finding the illumination using

instead of the formula using the dot product of two vectors. This also made one loop creating the gradient vector unnecessary. Deleting that loop also decreased the program’s runtime.

While completing this project, I learned that deriving in both x and y, can be quite challenging to figure out how to scale the numpy arrays to the same size when the derivation method returns only the central points’ derivatives. I’m still not completely sure that I scaled it 100% correctly, but with the granularity of the data provided it did not change much. If anything, it is simply shifted down one data point, which I’m unable to tell from simply looking at it. If there were only a few data points, more time and consideration would be put into ensuring the slopes calculated for x and y are not shifted accidentally.

Overall, this project was intriguing. I briefly advised a few other students on their projects which were also interesting projects. Although this project didn’t take too long to program, the majority of my time was spent on trying to illustrate concepts to others in an easy way. This explanation part was something that I’m not used to doing in my programming courses, as I’m typically assigned a project and just need to complete it. I’m still trying to find ways to help illustrate my problem and solution, and hope I will do so in a straight forward way by this time next week.

**References**

[1] M. Newman, Computational Physics, pp. 145-212, (2012).