3D Modelling for Urban Rooftop Solar Panels

Cooper Union 2016 Senior Project Final Report

Authors: Stephen Leone, Elizabeth Adelaide
Advisor: Sam Keene

<https://github.com/sml911/senior-project>

Abstract: Urban solar energy solutions are needed to increase storm resiliency, and to reduce energy wasted from long transmission lines running into the city. However, solar panel installation and current commercial solar calculation software costs are overwhelmingly high and deter the average landlord from adopting solar energy on the roof of their own building. The purpose of this project is to create an easy-to-use, accessible design tool which determines optimum placement of solar panels on an urban rooftop. A low-cost terrestrial LIDAR scanner was designed and will be used to determine the geometries of the rooftop and the surrounding buildings. This information will be processed to determine the amount of light that is incident on sections of the roof over a year. Optimal solar panel position and orientation is calculated by considering the shading factor, the airmass factor, and an approximation of the solar panel's angular response. This approach is explored in hopes that our design tool will give the average user the knowledge and confidence to be able to install solar panels and rooftop gardens without the need of unnecessary expenses.

Table of Contents

I. Introduction

Our project is a tool for designing solar arrays on urban rooftops. It allows landlords, neighborhood councils and other individuals or groups to map a site, and then use a software model to create an optimum array design based on energy production, cost and area. The design tool improves on solar array design techniques in dense urban areas, as well as increases the ease of adoption of alternative energy systems.

The ability to design efficient solar arrays in dense temperate urban regions, such as New York City, is a valuable capability which is increasingly being utilized. Energy solutions that are renewable, and that do not produce CO_2 are needed to prevent increased effects of global warming. In New York, CO_2 and pollutant production contributes to a heat island effect (Kim, 1992). The city is significantly warmer than surrounding areas, which leads to greater energy costs for cooling buildings. Replacing current energy solutions with solar arrays will reduce the heat island effect, and will reduce energy costs. Buildings that have their own solar arrays have off-grid power which allows them to keep power through power outages. This benefits tenants, especially those who are dependent on medical technologies. Local solar arrays are also at a closer distance to energy sinks, reducing transmission loss.

Heavily developed areas, such as lower Manhattan, present significant challenges to solar array design. Surrounding buildings create shaded areas that cause Many rooftops also feature smaller obstructions such as water towers, trees, ac units and sheds. An optimum design of panels must take into account these obstructions to determine the amount of energy output. In regions further away from the equator, which includes a large number of heavily developed vertical cities, optimum solar design is needed as the amount of solar energy available is limited.

Prior Art

There are several products on the market which aid in solar array design. These products do not address the specific needs of solar array design in dense urban contexts. However they do represent approaches to solar design from which our design works from.

Google Sunroof is a recently implemented free tool created by Google. Using satellite imaging, it calculates the energy available on every rooftop in a given neighborhood. As of today, a few neighborhoods are being piloted, mainly suburban areas. Sunroof can work accurately for suburban houses with limited external obstructions. However, it can only be

deployed slowly, requires a massive amount of processing power, and is not proven to work with complex roof geometries. It also relies on having up-to-date satellite images, preventing smaller groups from being able to map their own site. As the topology of dense urban environments like New York City change constantly, tools such as google maps will quickly become outdated for this purpose, leaving users with no control over which rooftop environments are still relevant.(Google, 2015).

SolarDesignTool is another solar array design tool. This tool is focused on architects and solar companies who design solar arrays in their profession. The product costs $500 per year, putting it out of the reach of a group that only wants to design for one building. The product requires the user to create accurate and precise architectural diagrams. In a setting where external obstructions greatly affect energy generation, this would require groups to survey not only their own building, but also other buildings. (SolarDesignTool, 2015).

Overview

Our work is divided into three sections, mapping, modelling and measurement. Mapping consists of the hardware and software to create a 3D point cloud scan of the site. Modelling consists of a software model which uses the 3D point cloud data to calculate the energy available to a given panel on the site, and then creates optimum designs based on energy production, area and cost. After mapping and modelling a test site, we will measure the energy available at several points on the rooftop. That data is used to verify the model, as well as to create improvements and changes if there are inaccuracies or practical limitations not considered.

II. Mapping

In order to accurately model the solar energy value of a rooftop, we must first get some reasonably accurate representation of its geometry. We have accomplished this by designing our own tripod mountable 3d LIDAR surveying tool which outputs a point cloud data file. This system is designed to be low cost and easy to use in addition to being accurate and efficient. The LIDAR scanner has 180° pan-rotational view and a 135° tilt rotational view and has a max scanning rate of ~100 measurements per second. The tool uses an adaptive technique to modify the scanning rate based on the distance of the most recently mapped points. This has the effect of taking more measurements to represent objects that are further away from the device to achieve a constant measurement density over all surfaces. The time that the scanner takes to map one half-sphere is normally 5-6 minutes and will measure around 30,000 points. The scanning time and density will however vary depending on the complexity of the environment.

Research and Method

The first problem presented during the research phase of the project was: how can we design an easy, inexpensive, and accessible method of obtaining a geometrical representation of a generalized nyc rooftop meant for the use of the non-professional? One suggestion was to use surveying equipment with gps capability to measure points on the building and construct a geometry from that. This approach requires relatively expensive specialized equipment and training which would quickly deter the average potential user from considering our tool.

Another option was to use the xbox kinect fusion to do simultaneous location and mapping (SLAM) by having a user walk around the rooftop and map the roof in real-time. This satisfies the requirements of being accessible and easy to use. After looking into previous SLAM work done with the kinect fusion, we determined the method to be less than ideal for a few reasons. First, the kinect does not work well in unbounded areas, posing a huge problem for any measurements taken in open air environments; second, the maximum range of the sensor is four meters which is not enough to detect large roof features or surrounding buildings; third, the SLAM problem is non-trivial and would likely be an entire senior project in itself. The mapping is not the focus of our project and we wanted to have a working prototype by the end of the first semester. Other automated mapping technologies such as sonar and radar have also been considered, but both would require high transmission powers, sensitive receivers, and advanced beam forming techniques to map distant objects and surfaces. Complicated roof

geometries might not be able to be resolved, and parts would be costly. These options were also rejected.

We concluded that LIDAR was the most viable technology to create a 3D point cloud of the rooftop. Because LIDAR measures distance using the time of flight of a laser pulse, the transmission power of the device does not need to be as high as the power of a spherical wave emitting device would be. State of the art LIDAR systems can create accurate point cloud map very quickly, measuring thousands of points per second. However, commercial 3D mapping terrestrial LIDARs fall in the price range of $10,000 - $100,000, far outside of our budget. Aerial LIDAR, though offering advantages in perspective and mobility, require specialized equipment (drones) or aircraft and extend even further beyond the price range. Companies who may own LIDAR systems or otherwise show an interest in the objectives of our project were contacted for possible collaboration; however, none of the companies responded to the requests. With these considerations in mind, we decided to use an inexpensive lidar unit and attach it to a pan-tilt system of our own design. Initially, the goal was to spend under $500 on this concept; however, the design ended up costing around $200. Although the precision and speed of this device is not as good as industry grade, the performance of the device is far better than what is required for the modelling stage of the project.

3D-Scanner Design: LIDAR

The Lidar Lite V2 by pulsed light was the LIDAR chosen for the scanner. Choosing this model over others was relatively easy because it is one of the cheapest, lightest, and well developed LIDARs in the market. Instead of using precision timers to measure the time of flight of a laser pulse like most LIDARs, the Lidar Lite V2 sends a pulse train and uses signal processing to correlate the transmitted and received signals thus finding the time of flight. This method is unique to the LIDAR market; it allows for the chip to be produced as inexpensive, small, and lightweight as it is because it doesn't need all of the extra timing components.

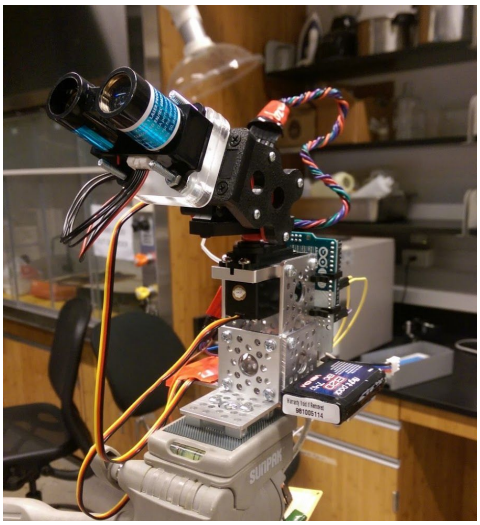

Figure 2.1: A photo of the functional 3D-Scanner with a focus on the Lidar Lite V2 unit

3D-Scanner Design: Pan/Tilt

Initially, the plan for the scanner did not include a pan/tilt device, but a turntable where the lidar and a tilting mirror controlled by servo motors would sit; the pan would be simulated by the turntable and the tilt by the mirror. After some research, the more cost-effective option became a pan/tilt camera/sensor system. Our system accepts any standard size servos for the motors. This model was chosen because it was the cheapest well-made pan/tilt system that supported the weight of the LIDAR.

Figure 2.2. A photo of the pan/tilt system and the servo motors used in the scanner

3D-Scanner Design: Servo Motors

The servo motors used are HS-485HB Servos this is an all-purpose servo with karbonite gears and reasonable torque and speed. Initially, we chose the HS-430BH servo which has a lower torque, higher max-speed, nylon gears, is cheaper, and runs at a higher operational voltage. This servo was chosen mainly because it was inexpensive and designed to work with the 7.4V LiPo battery pack. However, these servos quickly proved to be inadequate because they generated vibrations when stepping through small increments. The new servos operate at 5V and provide significantly less vibration during the scan.

3D-Scanner: Structural Components

We chose to use the servocity actobotics aluminum channels mounting hubs and L-brackets because it provided for easy servo and tripod mountability as well as flexibility in configuration. The only structural piece custom designed is the pan/tilt to LIDAR mounting adaptor. This was designed specifically to fit the LIDAR's non-standard dimensions and was laser cut from two pieces of acrylic then glued together using epoxy.

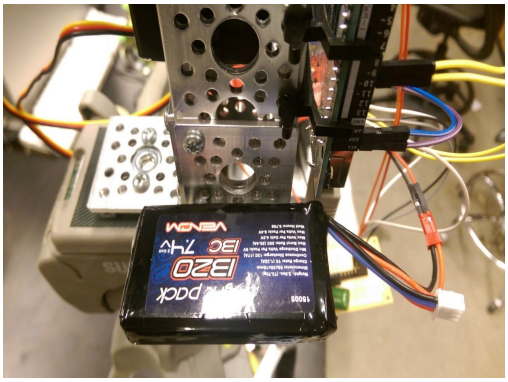

Figure 2.3 A photo of the LiPo battery pack mounted on the left-hand L-bracket

3D-Scanner: Battery Pack

The 7.4V LiPo battery was chosen simply because it is easily regulated to 5V and we had a few lying around. The power supply could be anything else from a AA battery pack to a single 9V. This design is flexible the sense that the left-hand L-Bracket shown in **Figure 2.3** can support any reasonably sized battery pack.

3D-Scanner: Microcontroller & Software

We chose the arduino uno for the microcontroller because of its versatility, accessibility, and large helpful community of users. The arduino code, which controls the data acquisition, scans through a range of angles in the pan and tilt directions. At each angular offset, the Lidar takes a distance measurement at each point, calculates the x, y, and z coordinates of that point, and sends those coordinates through serial to a terminal program on linux. Before the scan, the arduino goes through a series of handshakes which handle the transfer of data between the arduino and the terminal. This data could include an estimated number of data points, the angular offset of the measurement, scan speed, scan density, and other user inputs. The arduino also has a dedicated servo library as well as I2C support for the LIDAR which are benefits. The existing servo libraries were not able to adjust our measurement densities past one measurement per degree. Increasing measurement densities allows for more accurate scans at far distances. Decreasing measurement densities decreases scan times at near distances.

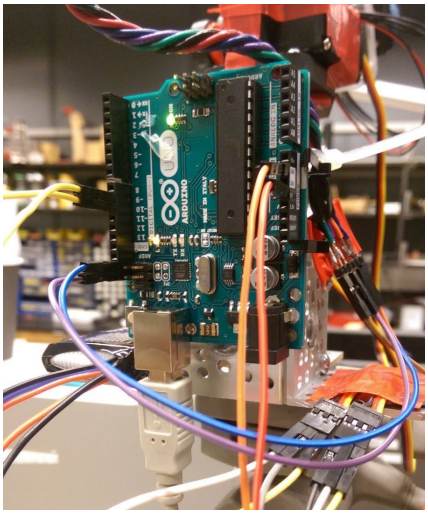

Figure 2.4 A photo of the Arduino board mounted on the scanner

The function of the terminal program is to provide a user interface, connect to the arduino, and output the data sent from the arduino into a text file in point cloud format. This program is written mostly in c but there is also some c++ for filestream support.

Scanner Initial Testing:

After completing the initial design of the LIDAR Scanner, we proceeded to testing. we started out in a hallway with lockers on the 6th floor of the NAB shown in **Figures 2.5 and 2.6**.

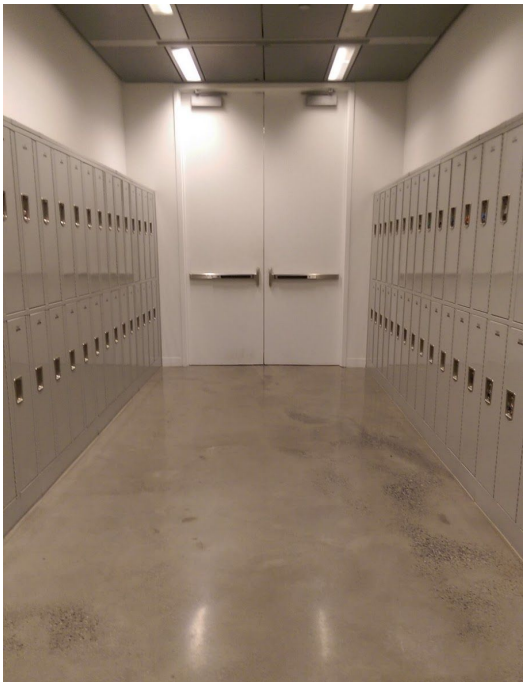

Figure 2.5 A photograph of the locker hallway testing area on the sixth floor of 41 cooper.

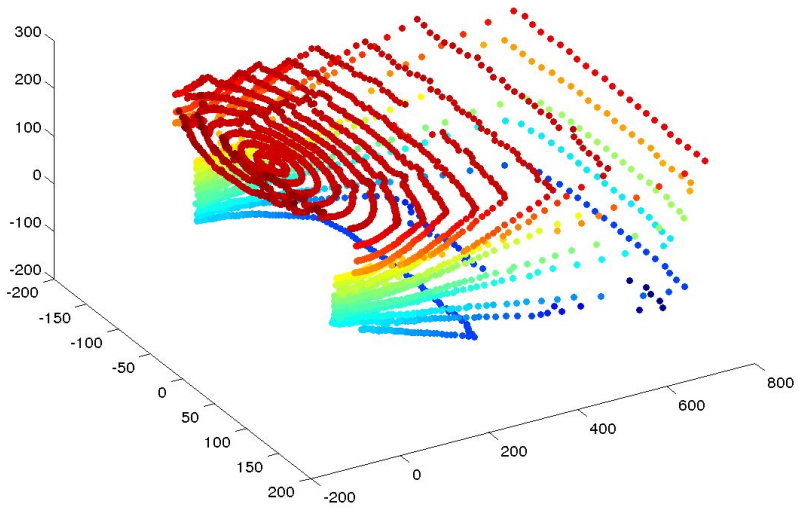

Figure 2.6 A point cloud visualization of the Locker hallway testing area.

we also measured the distance from the LIDAR to the far door with a tape measure to compare it to what the LIDAR measured. The tape measure read 102.9" and the LIDAR read 103.4" cm. This is a 0.5% deviation, well within the LIDAR's accuracy specification of +/- 5%. The next environment we tested on was the 5th floor window shown in **Figure 2.7**.

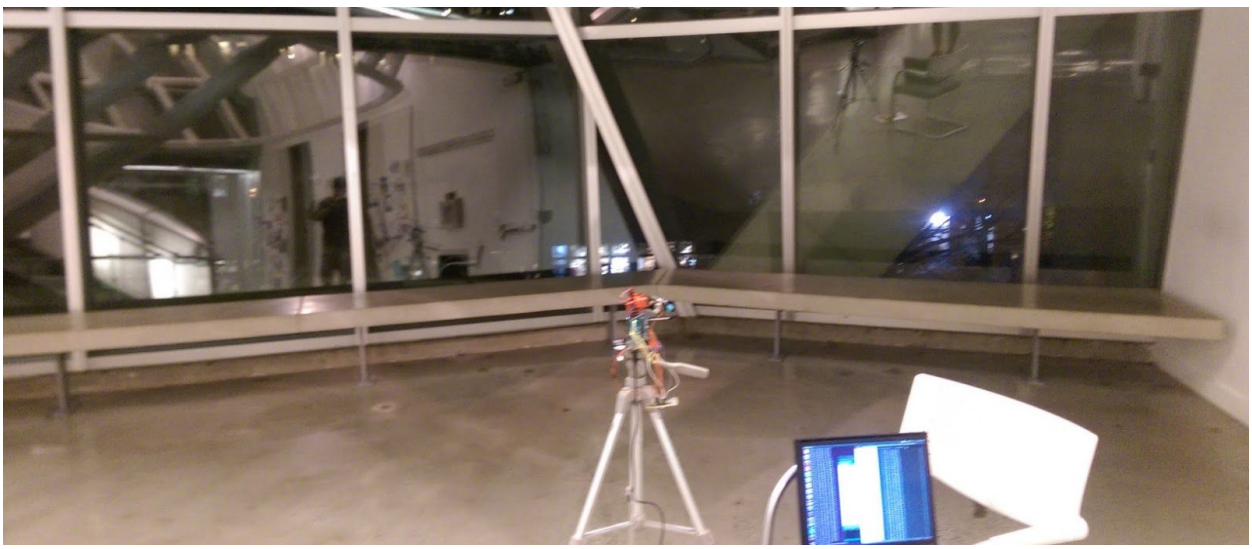

Figure 2.7 A photo of the 5th floor window testing area

The purpose of 5th floor window test was to see how the laser would behave when incident on glass. Would the pulse be reflected, refracted, or scattered? Looking at the point cloud in MATLAB, we could determine that, though there are a few outliers, most of the data points are clustered within 10m of the origin. This indicates that the laser pulse is scattered off of the glass which is good news for the purpose of detecting surrounding buildings with glass panels. This system would not operate correctly in the case that there is a glass enclosure on a roof. However, placing solar panels within a glass enclosure is generally not done because the sun will lose 15% of it's energy while travelling through clear glass (Shimazdu).

Figure 2.8 A photo of the Grand staircase open space testing area.

The third measurement we took was in the grand staircase of 41 Cooper. The purpose of this measurement was to see how the lidar performed in a large, open area. As in the 5th floor window measurement, there were a few outliers which made the point cloud difficult to visualize. This is taken care of in the modeling stage this measurement revealed that we did not have enough measurement density to accurately represent surfaces far away from the sensor. This spurred the idea of scaling measurement density proportionally to r in the phi direction and r*cos(phi) in the theta direction based on a moving average of r based on the jacobian of spherical coordinates. This would cause a fairly constant measurement density over all surfaces.

Scanner Improvement and Mapping Performance:

The initial testing phase was designed so that there would be ample time between subsequent measurements. This ensured that the motors were not moving too fast and that the LIDAR had time to stabilize its position before it took a measurement. Under these conditions, a full scan of around 4600 measurements took eight minutes and 40 seconds. The scan density here was one measurement per degree in the pan rotational axis and 0.2 measurements per degree in the tilt-rotational axis. This rate of measurement corresponds to about 10 measurements per second. In comparison, state of the art LIDARs can do over one million points per second at the pretty price of $123,915 (fltgeosystems). The LIDAR that we are using

has a maximum repetition rate of 100 meas/sec and an average rep rate of 50 meas/sec. Knowing this, we were able to optimize the scan speed against measurement integrity and arrived at a maximum sampling rate of 98 samples/s.

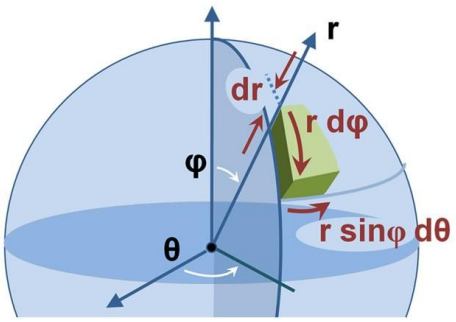

Figure 2.9 A representation of spherical coordinates which is the motivation for scaling the scan density by r^2sin(φ)

In addition to increasing the speed of the measurement process, we also increased the angular density of the measurements. In the initial testing phase, the angular density was limited by the provided arduino servo library function to 1 degree. After looking into the issue, we were able to find another function which allowed us to control the pan and tilt angles to a precision of one tenth of a degree; this resulted in 2.43×10^6 unique pan/tilt angular locations to work with. As was demonstrated in the locker hallway scan (**Figure 2.6**), points located at a smaller azimuth angles and points that are located closer to the scanner are measured at too high of a scan density. After maximizing the scan rate and density, we were able to scale the measurement rate according to a moving average of the radius. This moving average was implemented as a 10 tap FIR filter with a half-triangle window to minimize group delay. We use this moving average to scale the measurement density by the spherical coordinate jacobian, $r^2 sin(\varphi)$, where r is the distance from the lidar to a point on a surface, and φ is the azimuth (shown in **Figure 2.9**).

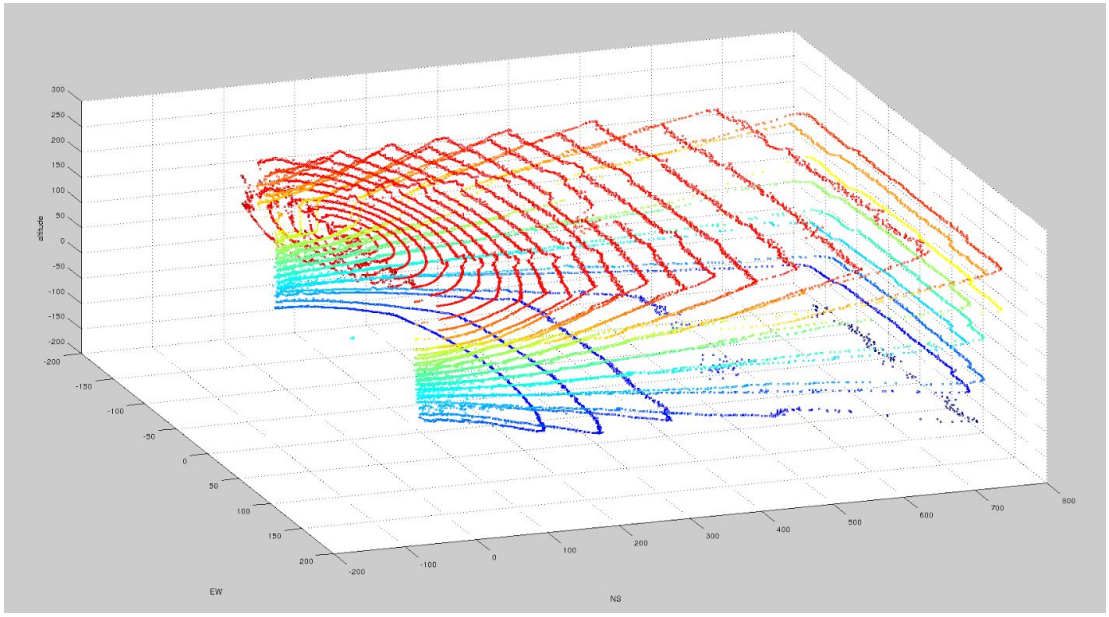

Figure 2.11 Point cloud of the locker hallway testing area with optimized speed and adaptive scanning density

After these improvements, a scan time in a test outdoor environment took 5 minutes and 42 seconds to take 32000 points. This corresponds to an average measurement time of 94 samples/s At this point in the design we considered the scan performance and quality to be sufficiently optimized on the software side and limited only by the quality of the physical components.

Shown above in **Figure 2.10** is a point cloud of the locker hallway with the finalized improvements made on the scanner. This can be compared directly with **Figure 2.6**. The scan density is much more uniform and dense than the first scan and was performed in less time. The point cloud from the original scan consisted of 4,500 points and the measurement took eight minutes and 27 seconds. The improved scan measured 23,538 points in 4 minutes and 27 seconds. The points that are taken at a high angle of incidence on the polished concrete floor (80° or greater) appear to be unreliable. However, this angle of incidence was never replicated on the ceiling because the lidar was closer to the floor then the ceiling during the scan. It should be noted that this may be a concern when measuring distant mirror-like surfaces at a high angle of incidence such as a reflective building.

The Point Cloud Data File Header

The software interface to the lidar generates a point cloud data formatted file and inserts a valid .pcd header so that the point cloud would be able to be used and manipulated by programs such as autodesk revit or through other programs using the point cloud library (pcl). A sample header is provided below, taken from one of our scans. The most notable field is the VIEWPIONT field; according to the PCD file format documentation, “VIEWPOINT specifies an acquisition viewpoint for the points in the dataset. This could potentially be later on used for building transforms between different coordinate systems, or for aiding with features such as surface normals, that need a consistent orientation. The viewpoint information is specified as a translation (tx ty tz) + quaternion (qw qx qy qz).” Currently, the translation is assumed to be (0 0 0) and the quaternion is calculated from the angular offset prompted at the time of measurement. (1 0 0 0) corresponds to an angular offset of zero and it is the default. The other fields are well documented on the pcl website.

```
# .PCD v.7 - Point Cloud Data file format
VERSION .7
FIELDS x y z
SIZE 4 4 4
TYPE F F F
COUNT 1 1 1
WIDTH 32190
HEIGHT 1
VIEWPOINT 0 0 0 1 0 0 0
POINTS 32190
DATA ascii
75.57 2.77 -36.57
77.33 3.67 -37.45
75.49 4.39 -36.57
79.03 5.45 -38.32
```

Figure 2.11 Example of pcl file header

Future Work

When mapping a rooftop from level of the rooftop, there will likely be obstructions which will not allow the user to create a map of the rooftop with one scan. In this case the user must take multiple scans. This presents an additional problem: How can we track the position and orientation of the LIDAR during the new scan with respect to the previous scan? An initial proposed solution to this problem was to use a compass/accelerometer/gyroscope to track changes in position and orientation between measurements. However, this was deemed an unreliable source of error and an unnecessary additional cost. We decided that the best way to manage multiple scans would be to have the user provide a rough estimate of the distance and angle offset to the software tool relative to the previous scan and have it use correlation techniques to merge the two point clouds together. Unfortunately, we did not have the time to implement this, nor did we encounter situations where moving the origin of the scanner between scans was necessary. Instead, we allowed for the angular orientation of the scanner to change between scans, this way two scans are able to map a full 360 deg sphere of visibility. In the future we plan to support automated point cloud merging to

Scan Tool Procedure

In order to take a scan, you will need the Lidar scanner mounted on a tripod, a computer using linux, and a usb cable. Our current scan procedure is to position the lidar facing north for the first scan, then rotate the lidar 180 degrees to face south for the second scan. This makes it easier to interface with the modeling tool. First power on the Lidar scanner by flipping the switch in the on position. Once the servo motors stabilize, position the lidar so that it is facing the north direction and ensure that the lidar platform is level. It is important to keep the lidar level during this process in order to keep the integrity of the measurements. This is easy enough to ensure on a tripod with adjustable legs and an embedded level. Connect the arduino to the computer using the usb cable and run the measurement program in a terminal. The orientation (in degrees) and the *.pcl filename is entered in the software tool's prompt. For the north facing scan, the orientation is 0, and for the south facing scan, the orientation is 180. You may take as many scans as you like provided you measure and input the proper angle during the scan. The output file that was specified will be used as input to the modeling software tool.

III. Model

The software model calculates the incident energy for a group of solar panels over a year. The MATLAB program calculates the amount of direct sunlight available for each panel, the incident angle of the light, and the amount of shading on each panel. The model does this by creating a surface from the mapper's point cloud data, and then uses solar geometry to determine the direction and strength of incoming direct light.

1. Surface Generation

The first part of the model creates a surface from the scattered points recorded by the mapper. These points are stored as x,y,z points. In order to perform calculations on these points, they must be mapped onto a regular discrete x,y grid, and they must be a connected surface. A regular x,y grid allows the program to perform a calculation on a discrete set of points. A surface with only one point per x,y coordinate is required as an assumption to create a solid surface. Otherwise, there would be a potential of overhangs or complex geometries which cannot computed or accounted for. Additionally, outliers must be removed from the mapping data.

The model assumes that the energy produced by a solar panel is proportional to a group of factors. The airmass, the lambert cosine, and the partial shading factor. The power produced by a panel can be expressed as:

$$P(t,\ L,\ S,\ \theta)\ =\ E * A(t,\ L)\ *\ S(t,\ L,\ S)\ *\ l(t,\ L,\ \theta)$$

Where P is the power at a given time, E is a constant efficiency factor, A is the airmass in kWHr, S is the partial shading factor, and l is the lambert factor. The power is a function of the time, t, the location, L, the site, S, and the tilt of the panel, theta. The model does not account for the more complex characteristics of the panel, which has a non-linear relationship between current and voltage. Due to the variation of panel efficiency and operation, as well as variations in charge controller and battery design, these factors were neglected. (Villalva, 2009) The model also assumes that direct light from the sun is the only major source of energy. While reflections and diffuse sources of light could be considered, the amount of energy available from them is negligible (Brine, 1983).

The program first imports all the data from the mapping. It performs an angular transform on the data, so that the data is rotated the same direction of the mapper. The program then removes all points that are greater than 50m away from the origin. This is done to remove any points that are beyond the range of the LiDAR. Any points beyond its range are assumed to be

errors. The program now has scattered points within a 50m sphere as shown in **Figure 3.1**.

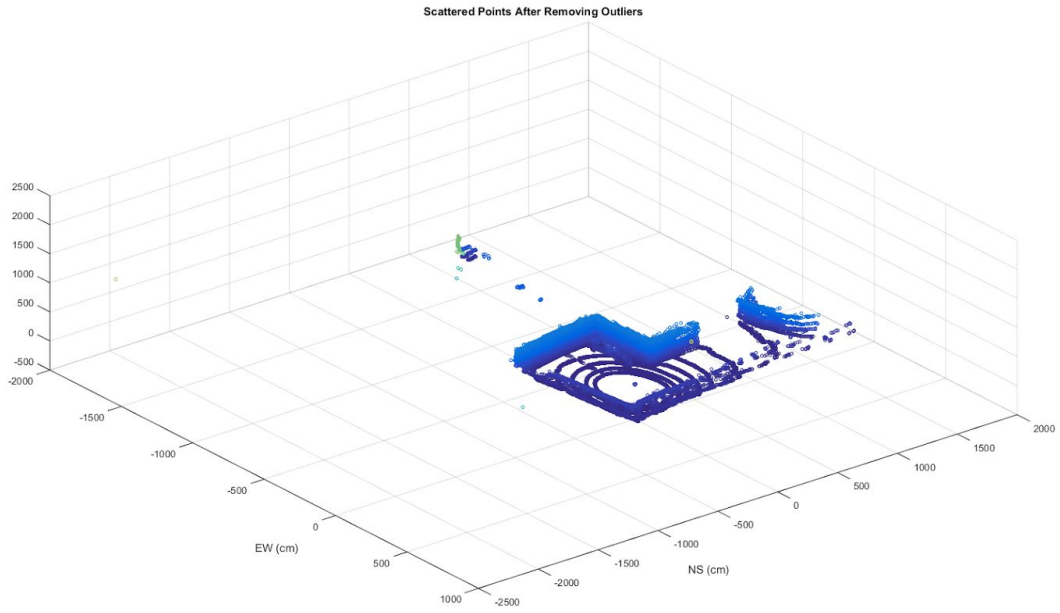

Figure 3.1 Scattered Points After Removing Outliers

The next step of the program is to map the points onto a regular grid. The first step is rounding the x, y coordinates to the nearest integer. The z coordinate points are preserved. The next step is to create a surface with no overlaps or holes. A regular x,y grid is created that contains all the rounded coordinates. The maximum z value for each coordinate on this grid is then found. If the coordinate does not contain any points its z value is assumed to be zero. A surface is then generated from these maximum points. The reason why the maximum heights are taken, rather than the average height, is that the surface will consist of buildings or obstructions that are assumed to be solid and perpendicular to the ground. Any point that mapper registers is assumed to be a part of an object which extends to the ground. This does exclude fences and overhangs. The surface found from the maximum points is shown in **Figure 3.2.** The surface has several peaks and isn't very smooth. However it does represent the geometry of the building and surrounding obstructions surrounding the site.

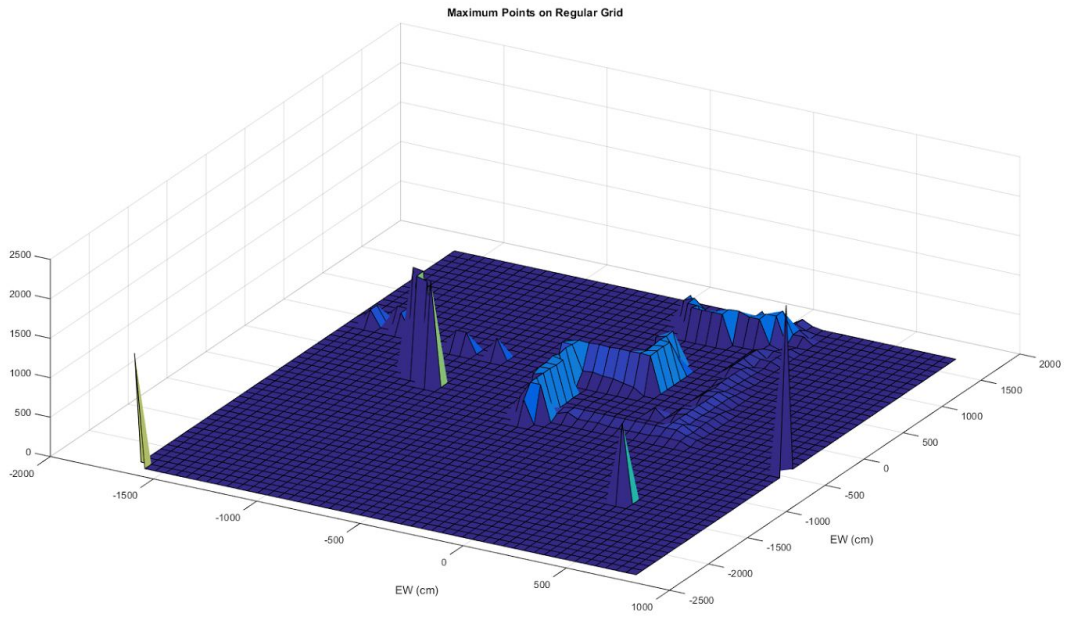

Figure 3.2 Maximum Points on Regular Grid

The next step is to remove the peaks, and to connect surfaces that are close together but are close enough to be the same wall or feature. Those surfaces need to be connected for places where the mapper has lower scan density. Additionally, The surface are made more solid to account for panels that are placed further away from the origin. The program calculates the average distance between points on the surface that are non-zero. For each point on the grid, the program checks a radius of the this distance for any point that has a higher altitude. If there is a checked point with a higher altitude, then the altitude of the point is set to that value. If the radius only contains points of altitude zero, the point is an isolated spike and the altitude of the point is set to zero. This process smooths the data, and removes outliers. The result of this process is shown in **Figure 3.2.** The surface has removed the isolated peaks. Additionally the taller feature of the roof is now seen as a more solid object. The areas of the features is also increased by this process. While this may lead to an overestimate of the amount of shade, it is assumed that times when the panel is about to be in shade or has just left being in shade that the panel is in soft shade which also reduces the incident energy available for the panel.

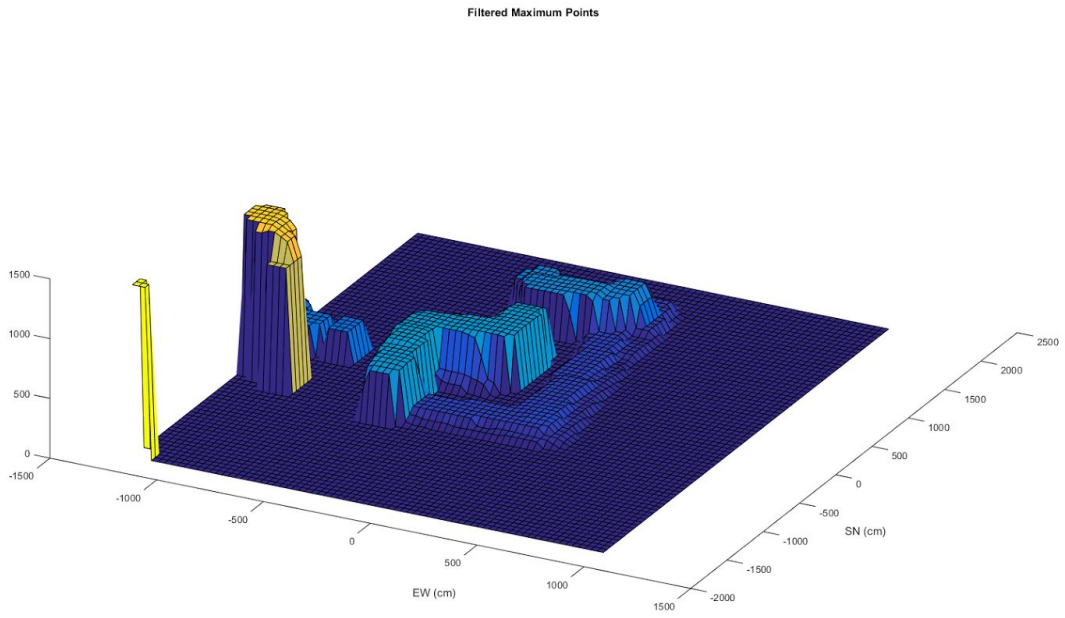

Figure 3.3: Filtered Maximum Points

For comparison, the site is shown in **Figure 3.4**. The round feature (an elevator shaft) is accurately shown, as the walls of the building. Our LiDAR was unable to detect any surrounding buildings. The model and the LIDA also created a surface for the guardrail which surrounded the accessible portion of the roof. The model will treat that as a solid object, which will underestimate the amount of sunlight available. However, this is only a problem when the sun is low in the sky, which is already where the least amount of energy is available. The tall feature in the SE corner is an outlier that was not removed by the program. This will also cause underestimation of the amount of light available.

Figure 3.4: Site for Model

The surface generation requires a significant amount of computing time. For larger grid densities, the surface might cause MATLAB to run out of memory. There was little difference in the overall output for grids larger than 100x100. The finer the grid, the more details are brought out, and the amount of underestimation from outliers and small surface is reduced.

2. Calculating Incident Light

After the surface is generated, the program can calculate the amount of energy incident on each panel. The program first calculates the position of the sun at each hour. It then determines if there is any shading on any part of each panel. This is the main part of the program. With information on the incident energy on each panel per hour, an array can be easily designed with the highest energy output.

The panels are positioned on a user defined space where the solar panels could be placed. This could be improved by detecting where the roof is, however that is a difficult problem in itself. The panels have a length and height which is mapped onto the discrete grid. For a fine grid, the panel may occupy several points. For a coarse grid, each panel may only occupy one point. The more points each solar panel occupies, the more accurate the partial shading factor will be. The partial shading factor accounts for when sections of the panel are in shade, which severely decreases the amount energy generated by a panel. The panels are assumed to be flat and at ground level for this part of the program. Accuracy could be improved by accounting for the height of the mounted solar panels, as well as the hypothesized optimum tilt of the panel. Currently the tilt of the panel is calculated for its optimum position after calculating the incident energy and shading.

The program calculates the zenith and azimuth of the sun at each hour. These determine the position of the sun. For points where the zenith is greater than 90^{o} the sun is beyond the horizon and the calculations are ignored for those points. The program uses a pre-written MATLAB function, *sun_position* to calculate the azimuth and zenith. (Rino, 2014)

The program then creates rays which originate from each point on each panel in order to determine if a panel is in partial or complete shade. These rays are first calculated as parameterized rays in the direction of the zenith and azimuth originating from each point of each panel. The parametric variable, *t*, is represented in matlab as a discrete set of equally spaced values. The length and spacing of *t* is determined by the limits of the surface. The parametrized rays are represented as *x,y,z* coordinates. Those coordinates are then mapped onto the discrete *x,y* grid of the surface by rounding the rays' *x,y* coordinates to the nearest integer. The result of this calculation is that the height of each ray at each point can be compared directly to the height of the surface at each point. Comparing the heights of the rays which extend towards the direction of the sun allows the program to determine if that point on the panel is in shade. The panel is in shade if at any point along the ray, the height of the ray is less than the height of the surface at that point. This indicates that the path from the sun to the panel is interrupted by the building or feature. This part of the program assumes that the roof is relatively small, so that the position of the sun does not change across the surface being calculated.

Figure 3.4 shows the rays incident on the surface for a group of points. Squares on the surface that are black, as well as rays that are black indicate that there is an interruption somewhere along the ray. For this site, the guard rail is causing shade to be detected where there might not be any, as seen by the rays which do not seem to intersect anything obviously. They are interrupted very close to where they intersect the surface. This issue can be solved by manual noting where objects such as guard rails are, or by employing a more complex surface generation process.

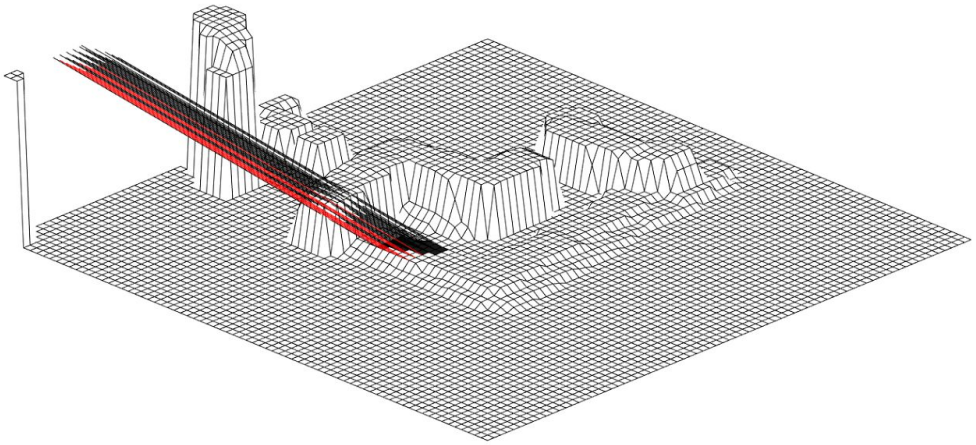

Figure 3.4 Rays incident on Surface

The program can then calculate the airmass and the partial shading factor for each panel. The airmass factor represents the amount of solar energy that is incident on a given surface at a given hour (Kasten 1989). It is a result of the amount of air that a given ray of light travels through to a point on the earth. The greater amount of air, the less light reaches a location. The amount of air that a ray passes through is determined by the zenith angle. When the sun is lower in the sky, there is a greater mass of air. The energy of light available is determined by the airmass, and is empirically given by the formula:

$$A = 1.35 * \frac{1}{1.35} exp(sec(Z)) \; [kW/m^2]$$

Where *A* is the airmass, and *Z* is the zenith angle. 1.35kW/hr is the ideal maximum energy. The airmass is calculated for each hour. It does not vary for each panel, because it is assumed that the zenith is constant across the rooftop. (Bird, 1985)

Figure 3.5 shows the energy incident due to airmass over one year for the site in Manhattan. The airmass varies over every day. **Figure 3.5** shows the average power per day. The maximum value of available energy occurs at the summer solstice, the lowest point at the winter solstice. Over a single day the available power ranges from 0kWHr to .9kWHr at noon on the summer solstice. The airmass is assumed to be the dominant factor in the available energy for a given panel. This approximation can be made more accurate by accounting for the probability of a day being cloudy or rainy, and the impact of indirect light.

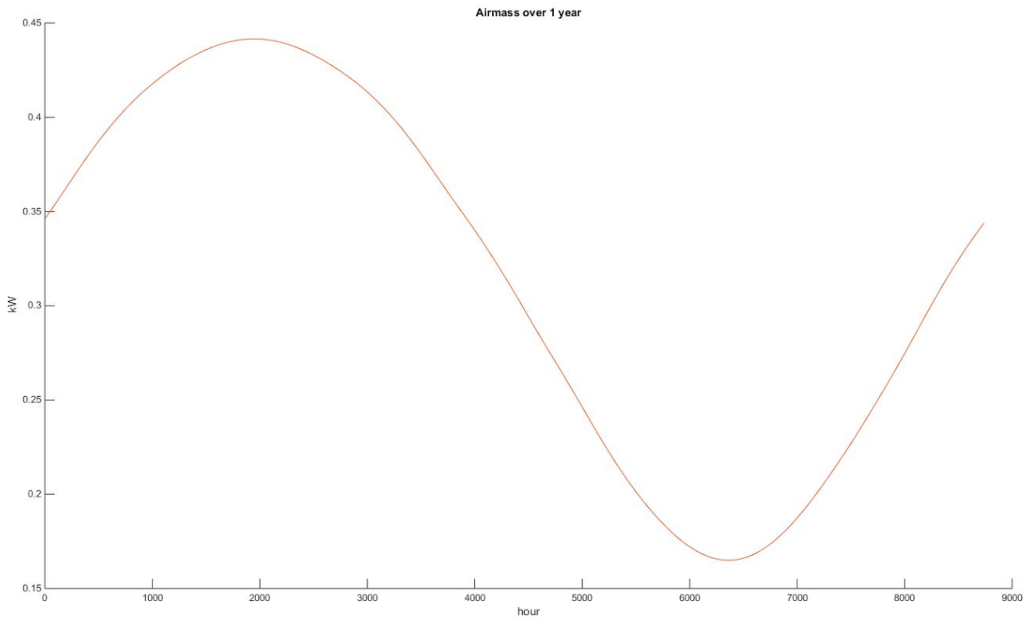

Figure 3.5: Airmass energy over 1 year

The partial shading factor is based on the impact of shade in energy generation for the solar panel. For a given amount of incident light, even a small amount of shade on a corner of the panel will greatly reduce the output energy of the panel. This loss in energy is the result of a cell acting as a voltage/current sink instead of a source (Patel, 2008). The impact of shading on a given cell is dependent on the arrangement of the cells on the panel, i.e. parallel or series arrangements. While the influence of shade can be very complicated, depending on where on the panel the shade is and if the shade is soft or hard, the program assumes that any energy the panel would produce is reduced by 80% under the influence of partial shading. This assumption is used as the panels' occupy a finite number of points on the grid. Using more complex models of partial shading require knowledge of the shading on each cell of the panel (Nguyen, 2015).

Figure 3.5 shows the partial shading factor over a year for 16 panels. Each color represents a different panel arranged on the roof. The factor was calculated for each hour and then averaged to each day. Over the year, each panel will be in different amounts of shade over the day. A more complex calculation of partial shading would have a more continuous curve for each panel, however it would still be largely discontinuous due to the nonlinear nature of shading. The figure shows that shading is the major factor differentiating the energy a panel will produce over a year. There are two groups of panels, which will have very different efficiencies. The panels also receive less shading during the winter due to the arrangement of the roof.

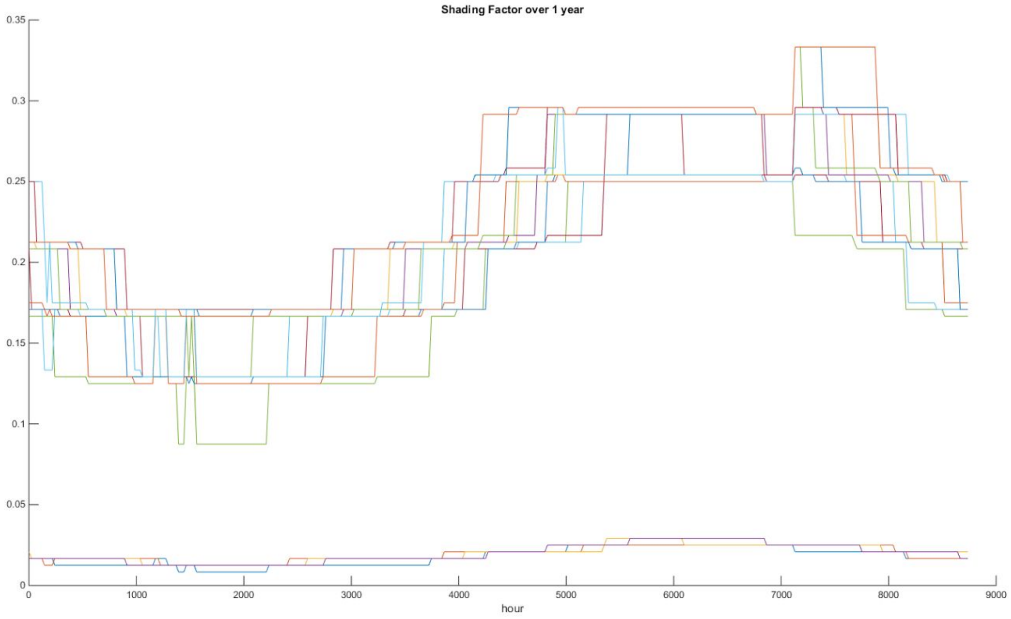

Figure 3.5: Shading Factor over 1 year

The airmass and shading factor are calculated in small groups of hours. This is to reduce memory usage which can become very large, because the calculations numerically go through every panel, every hour, and every point along a ray. The program can adjust the number of calculations done per hour which linearly increases computational time. Increasing the number of panels and the number of points per panel also increases the computational time greatly. Increasing the number of grid points is the most significant factor, as it influences the number of ray points to calculate which is the most complicated part of this section of the program.

After calculating the shading and airmass factor, the program approximates the tilt of the panel. For non-tracking panels, the tilt of the panel is mainly determined by the latitude of the site. Non-tracking panels do not rotate with the sun. Some sites will manually move the panels twice a year to optimize for winter and summer (Bird, 1986). The program currently assumes that the panel remains static throughout the year, and that the user wants the highest average power over the entire year. The tilt of the panel can be improved by accounting for the variations in shading over the year. The optimum tilt can be found iteratively by calculating and recalculating the average energy over the year until the highest point was found(Yudav, 2013)). However, to save complexity the program simply finds a weighted average of the energy approximated by the airmass and shading.

$$\theta = 90^0 - \sum_i m_i Z_i / \sum_i m_i$$

Where θ is the tilt of the angle from the ground, m_i is the product of the airmass and partial shading factor for each hour, and Z_i is the zenith at each hour. For this site, the optimum tilt was approximated to be 40^0, this is a slightly greater tilt than the suggested 33.5^0 for New York

latitudes (Hofierka, 2009). This is the result of the weighted shading factor, which decreased the available energy during the summer, requiring a greater tilt for a higher average power output.

After knowing the tilt, the Lambert Cosine factor can be accounted for. The Lambert Cosine factor is a model of the angular dependency of a solar panel to incoming light. It is the simplest model given by the equation:

$$L = cos(\theta - Z)$$

Where *L* is the Lambert cosine factor, θ is the tilt and *Z* is the zenith angle. For a given amount of incoming light, calculated by the airmass factor, a panel produces energy dependent on the angle through this relation.

Figure 3.7 shows the Lambert cosine factor over a year for 16 panels. The factor is averaged over each day. There is slight variation between each panel, depending on the specific angle that was approximated to be optimum for the panel. The discontunuity of the graph is a result of discretization of the zenith angle in the equation. Because the program only calculates the zenith once per hour, the length of one day jumps as it shortens and lengthens. For the lambert factor this changes the length and range of the zenith. For airmass and partial shading the zenith angle is a less direct factor, so it is not evident in the output.

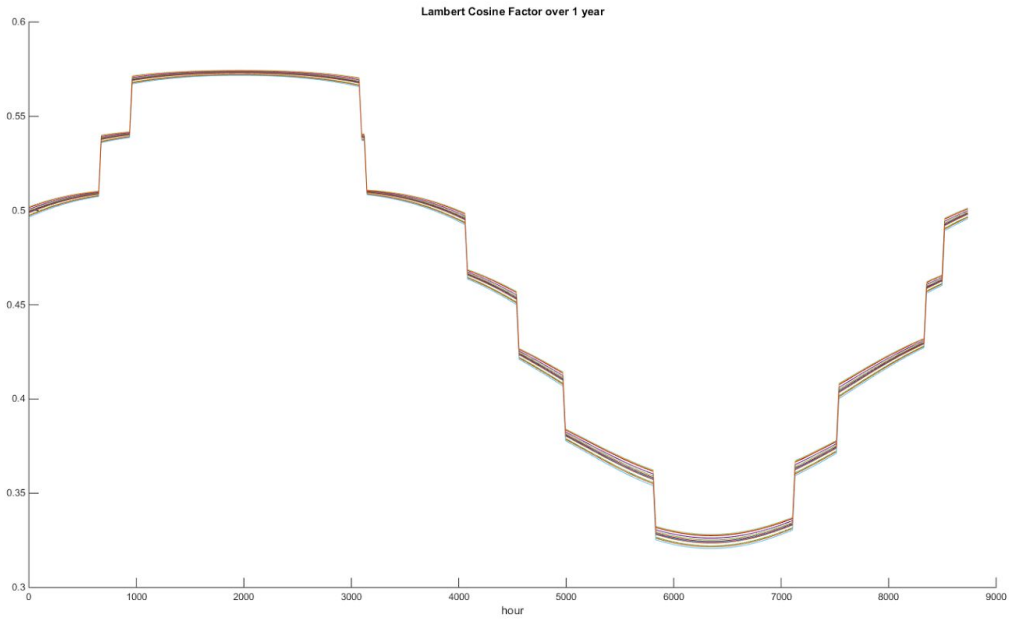

Figure 3.7 Lambert Cosine Factor over 1 year

With the airmass, lambert cosine and partial shading factor found over each hour in a year, the energy for a given panel can be calculated. The energy produced by a panel is proportional to the product of the given amount of energy of incident light, the angle of the light, and the impact of partial shading (Villalva, 2009). There are other factors that could improve the

model. The model assumes that the efficiency of the panel is directly proportional to the intensity of the incoming light. This assumption ignores the nonlinear response of panels, especially in less light and indirect light, and it ignores the gradual loss of panel efficiency within the first years of use (Jordan, 2012). However, the model does predict the impact of shading over a year, which is a major obstacle in solar panel design in dense urban areas.

Figure 3.8 shows the average power of the 16 panels over one year on the surface. There is an easily identifiable group of red panels, which produce the most power over one year. This graphic can inform a designer or installer where to install panels to avoid the nonlinear impact of partial shading. For this example only a small section of the roof was modeled, The model can be expanded to large sections of rooftops. **Figure 3.9** shows average power over each day for a year for the panels. It shows the trends in power generation over a year, with a maximum in summer and a minimum in summer. There is one panel which has significant shading throughout the year, which generates considerably less power, in orange. The rest of the panels show the same general trend. In winter, the panel represented in purple generates significantly less power in winter due to shading effects. While this impacts the total power over a year, it is also a concern to have irregular power generation during seasons with higher energy costs or requirements.

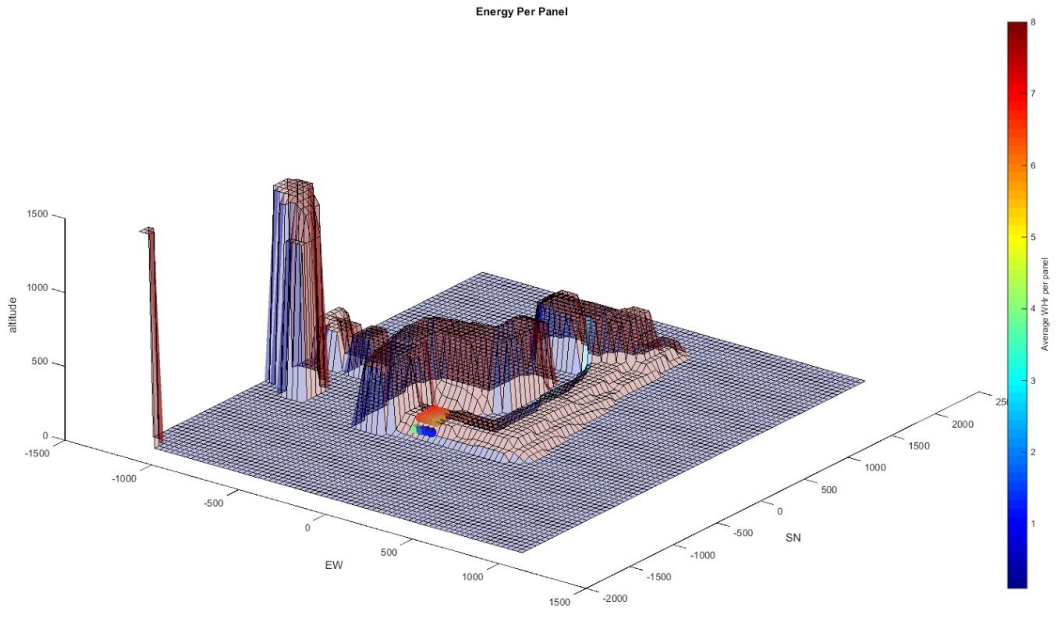

Figure 3.8 Average power per panel over 1 year

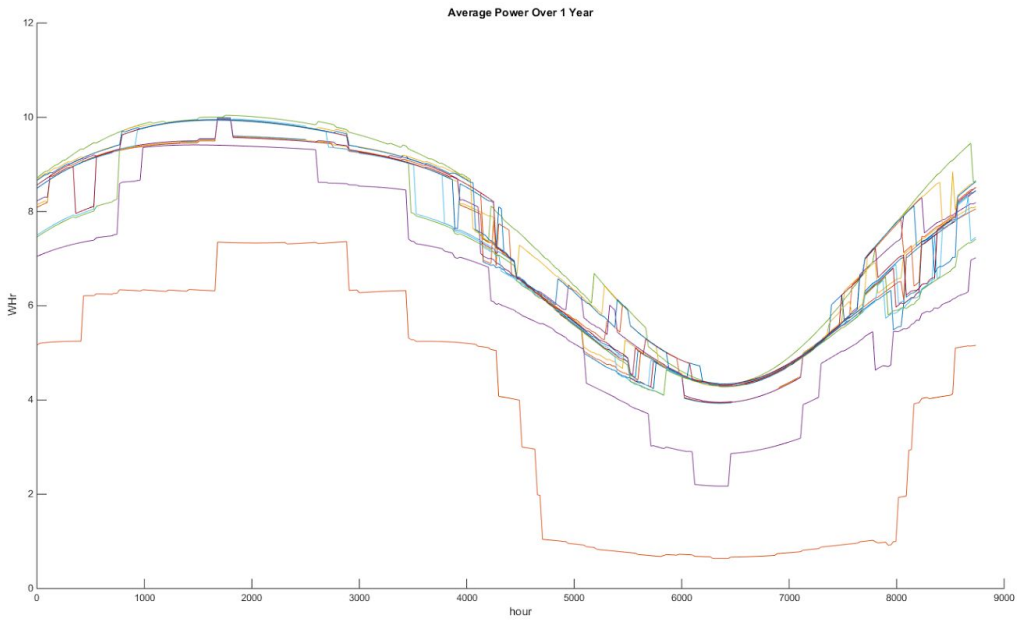

Figure 3.9 Average Power Over 1 Year

The software model predicts the power output of a group of given panels over a year. The panel allows a designer or installer to make informed decisions on the impact of shading over a year, and to predict the optimum tilt angle that the panel should have. The model creates several visualisation tools to assist in this process. Improving the model involves considering more effects, and allowing the specific characteristics of panel, charge controllers and batteries to be considered. The software also requires an accessible user interface to make design easier.

IV. Measurement

1. Introduction

In order to verify the results of the model,a setup to monitor the power output of a physical panel. Then the power estimated by the model can be compared, and any issues can be found. Due to restrictions on time and space, the panel could only be set up for a few days at a time on specific sites. Ideally, a large variety of sites could be tested over months or years to verify the model. Alternatives such as small-scale tests or lab simulations were also considered, however they were either too costly or did not represent the problems of shading in urban rooftops. There was also not significant existing data in a site where partial shading was an issue, and where the site could be mapped or simulated.

2. Hardware

For our measurements, we used a PV array with an O.C. Voltage of 20.4V and a S.C current of 2.53A. in order to monitor the power of the array, we created a data logging circuit which measured the power output of the solar panel over the course of the day. The efficiency of power transfer output from the solar cell depends on both the amount of sunlight falling on the solar panels and the electrical characteristics of the load. As the amount of sunlight varies, the load characteristic that gives the highest power transfer efficiency changes, so that the efficiency of the system is optimized when the load characteristic changes to keep the power transfer at highest efficiency. This load characteristic is called the maximum power point and MMPT is the process of finding this point and keeping the load characteristic there. Electrical circuits can be designed to present arbitrary loads to the photovoltaic cells and then convert the voltage, current, or frequency to suit other devices or systems, and MMPT solves the problem of choosing the best load to be presented to the cells in order to get the most usable power out. This motivates the need for a data-logging circuit which can measure the power output for different load characteristics.

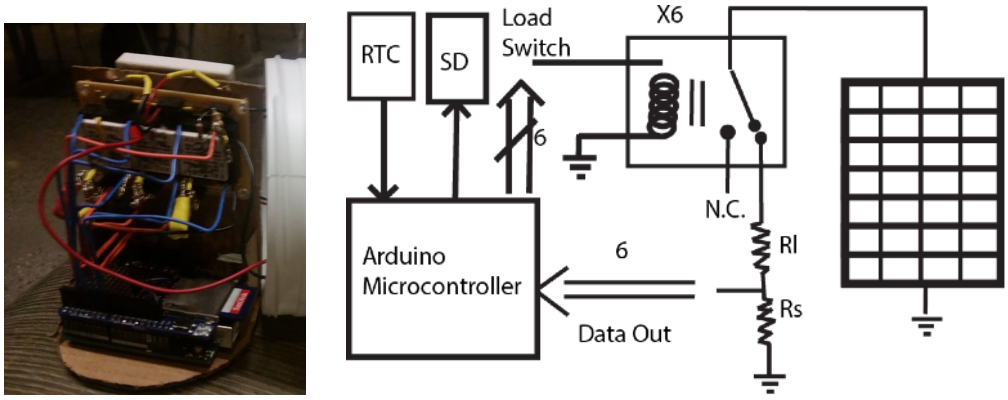

Figure 4.1 Data Logger Circuit and Schematic

The datalogger consists of an arduino microcontroller, a data logging shield, and a switched load as shown in **Figure 4.1**. The data logging circuit was initially designed to switch between six loads. This was later reduced to four loads because of a shortage of analog input pins on the arduino while using the data logging shield. The data logger measures the solar panel output power using a voltage divider and an analog input to the arduino referenced at 3.3V. The arduino writes the data, along with a timestamp provided by the real time clock (RTC) and the load number, to an SD card in csv format. The data on the SD card could then be analyzed by any spreadsheet software tool.

The next step in the design was to then choose the loads to use. The way that this was decided was to measure the maximum power point of the solar panel we planned to use for testing in a laboratory environment. The max power was multiplied by a large number close to 100 which set the maximum allowable power for the sensors. This max allowable power ended up being 1W meaning if the panel output over 1W, the analog input would saturate at 3.3V. The optimum load was also measured at the max power point and a range of loads with real laboratory resistor values were chosen. The resistors that we used for testing had a 1/4W power rating so we ended up using 5-10 resistors for each load. After our first outdoor testing experience, our resistors burnt out and we realized that we needed to reevaluate the choices of resistor values because the output power was higher than expected. During our subsequent tests, the resistors were chosen such that the sensor's saturation voltage was close to the open circuit voltage labelled on the solar panel. However, the voltage divider was still did not attenuate the voltage enough so that for the lowest impedance load, the maximum detectable power output was 5.6W.

This device was originally powered by an alkaline 9V battery; however, we found that these batteries only last about 11 hours while powering the arduino. Because we wanted to collect data over multiple days, we switched over to a LiPo rechargeable 9V which we were able to recharge using the power generated by the solar panel. The solar panel would only be in battery charging mode while there are no measurements being taken which is realized using a relay. This ensures that the charging operation does not interfere with the measurement integrity. Measurements are taken every ten seconds for cycling load profiles. Since there are four load profiles to cycle through, each load profile will have power output measurements taken every 40 seconds. This makes for a good balance between data point density and power consumption.

3. Results and Analysis

The panel was measured for three days. The hardware used to measure the power output only detected power up to 5.6W. Although the maximum output of the panel is 20W, the hardware was able to represent the panel's performance during partial shading, and when the incident energy was low, near sunset and sunrise. This allowed an evaluation of the model's prediction of partial shading, and its accuracy in low light. Unfortunately the hardware did not

allow a comparison for high direct light, however this is the most researched part of solar panel performance, and there is a large amount of recorded data (RReDC, 2016).

Figure 4.2 Solar Panel Installation on the Foundation Building

Three days of data were recorded on a building in lower Manhattan. The panel was placed on the roof, and data was sampled several times a minute. There was an error which affected the majority of the third day's data, however the remainder of the data was usable. The rooftop was mapped and entered into the software model. **Figure 4.3** shows the recorded data compared to the predicted power of the panel. Each peak of data occurs at noon of each day. The data, in blue, reaches a plateau at 5.6W, which is when the datalogger overloads in data. When the panel outputs less than 5.6W the data is a reflection of the power of the panel. The model's predicted power is scaled to the data by assuming that the panel will output 20W (the rated power) at the maximum ideal power predicted by the power, 1.35kW/m^2.

The software model was run several times with changes to the number of calculations per hour and the density of the grid. Because of the significantly shorter time of three days, and only having to calculate power for one panel, the complexity of the program could be increased significantly. Increasing the number of calculations per hour allowed a better comparison to the recorded data. Increasing the density of the surface grid didn't show significant changes in the timing of partial shading, however it did allow for the panel to occupy more points on the surface. This meant the model could detect shading on different sections of the panel, giving a better picture of the effect of partial shading.

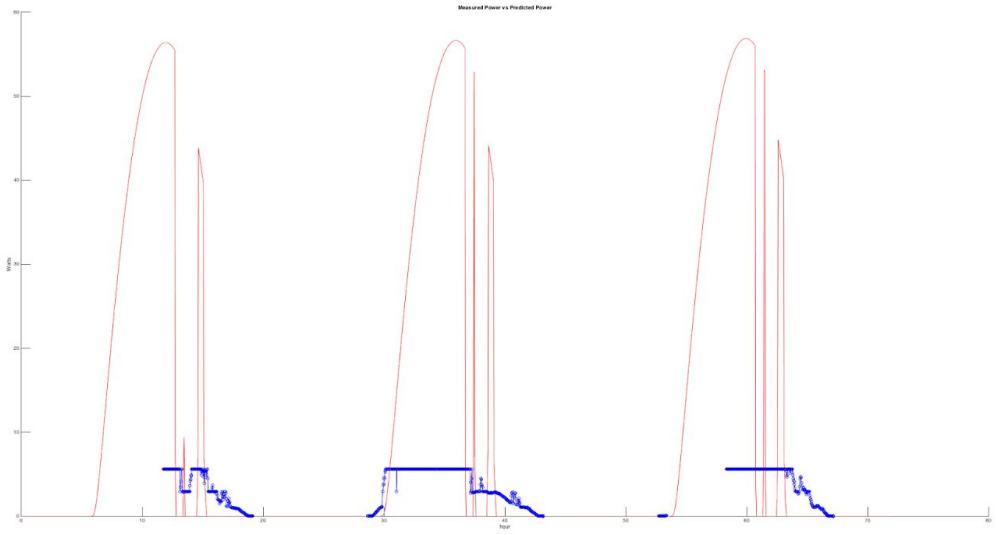

Figure 4.3.4 Comparison of model to data from rooftop panel

The model predicts the dropoff in power that is seen in the data on day 1 and 2 slightly after noon. The data also reflects the absence of partial shading conditions in the early half of each day. The model does not accurately show the gradual dropping off of power at the end of the day. This is likely due to the impact of indirect light. The data also doesn't reflect the sharp spikes in power when the model predicts there is no shading. These sharp spikes may be the result of slight differences between the surface and the physical sight, and the partial shading of specific cells that were not accounted for in the model.

Figure 4.5 shows a detail of the second day. The model doesn't accurately predict the panel's power towards sunrise. The panel's output at that time is due to indirect light that the model does not account for. This will result in a minor underestimation of the power produced by the panel. The model also predicts that the panel will enter into shade sooner than the data shows. This is due to the discretization of the surface and the overcompensation that was made in the surface generation process in order to have continuous features. The data does have several spikes after entering shading that may correspond to the predicted spikes, however the panel does not output nearly as much power as predicted during these spikes. This creates an overestimation of the amount of power produced. It is also difficult to determine if there was any influence of cloudiness or debris without much more data.

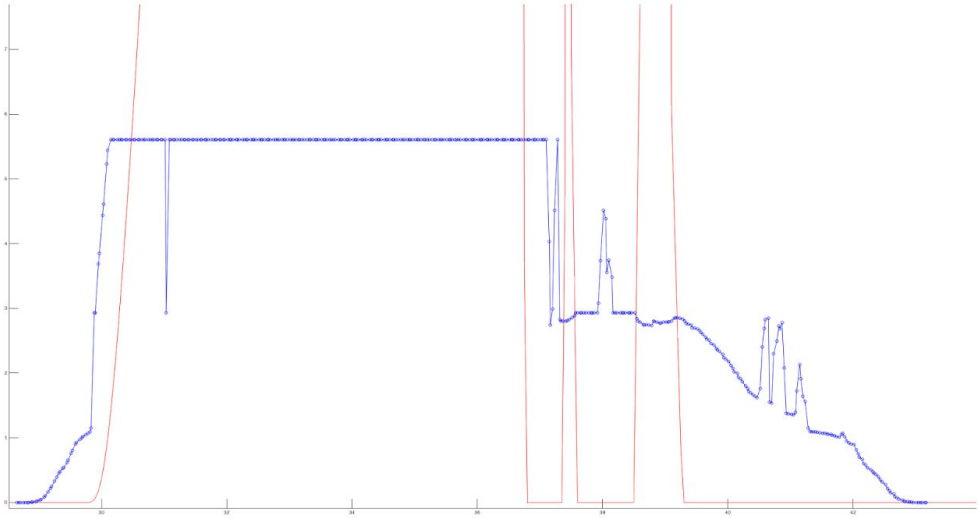

Figure 4.5 Second day of data compared to model

The model showed an ability to accurately predict when the panel would be in shade. The model also matched the rate of change of power during sunrise, where there was no shading. The model would require more testing on several different types of panels in various sites to completely confirm its precision and accuracy in estimating power at a given time. Additionally, more work could be put into modeling the panel for indirect light and in other low-light situations.

V. Conclusion

Our Solar Panel Placement tool helps the user gain a valuable perspective on the effects that solar panel position and orientation have on annual power output. We have introduced a low cost terrestrial LIDAR-based system to map the rooftop's geometry quickly, accurately, and effortlessly at a rate of 93 meas/s. Surrounding buildings at a distance of up to 40m will also be included in the scan. The modelling process developed requires light calculation compared to other, more accurate point cloud to mesh methods. Under certain assumptions, the smoothing performs well for the purpose of simulating a shading obstruction. However, improvement can still be made on the smoothing for specific cases such as the railing on the foundation building rooftop. The shading factor, the airmass factor, and an approximation of the angular response of a solar panel are taken into account when calculating the solar Energy. The power output is calculated over the course of each day for a year at each considered location and a visualization or power heat map is created as a final output for the tool. Measurement data was then used to scale the calculated power with respect to the efficiency of our solar panel.

VI. Appendices

Appendix A: Parts Cost Sheet

Supplier	Part	Count	Unit Price	Price
RobotShop	Lidar	1	$114.86	$114.86
	Shipping	1	$0.00	$0.00
	Sub-Total			$114.86
ServoCity	HS-430BH Servo	2	$13.49	$26.98
	Direct Drive Pan & Tilt	1	$19.99	$19.99
	3.00 inch Aluminum Channel	1	$3.99	$3.99
	90 Degree Quad Hub Mount E	1	$5.99	$5.99
	1/4-20 Machine Screw Hex	4	$0.08	$0.32
	Arduino Channel Snap Mount	1	$1.89	$1.89
	90 Degree Single Angle Channel Bracket	1	$1.59	$1.59
	Shipping	1	$6.99	$6.99
	Sub-Total			$67.74
Amazon	50in Tripod	1	$11.95	$11.95
	Tax	1	$1.06	$1.06
	Shipping	1	$0.00	$0.00
	Sub-Total			$13.01
Atomik RC	Venom 13C 2S 1320mAh 7.4 LiPO Battery	4	$6.97	$27.88
	EV-Peak V3 2-3 Cell LiPO Battery Charger	1	$6.99	$6.99
	Shipping	1	$0.00	$0.00
	Sub-Total			$34.87
ServoCity	HS-485HB Servo	2	$16.99	$33.98
	90 Degree Quad Hub Mount C	1	$5.99	$5.99
	90 Degree Single Angle Channel Bracket	3	$1.59	$4.77
	Shipping	1	$6.99	$6.99
	Sub-Total			$51.73

Supplier	Part	Count	Unit Price	Price
Amazon	Arduino, Batteries, and Datalogging Shield	1	48.7	48.7
	Tax	1	0.54	0.54
	Shipping	1	0	0
	Sub-Total			49.24
Amazon	EBL Rechargable 9V Batteries	1	23.99	23.99
	Tax	1	0	0
	Shipping	1	0	0
	Sub-Total			23.99
	Total			268.84

VII. References

Works Cited

Kim, H.. “Urban Heat Island.” *International Journal of Remote Sensing.* 13:12. 1992.

Google. *Sunroof.* FAQ. 16 December 2015. Web.

SolarDesignTool. *SolarDesignTool.* Features. 16 December 2015. Web.

Nunez, Manuel. “The Energy Balance of an Urban Canyon.” University of British Columbia. December, 1974.

Bird, Richard and Riordan, Carol. “Simple Solar Spectral Model for Direct and Diffuse Irradiance on Horizontal and Tilted Planes at the Earth’s Surface for Cloudless Atmospheres.” *Journal of Climate and Applied Meteorology.* 25. January 1986.

Rino, Charles. *Full Vectorization of Solar Azimuth and Elevation Estimation.* MATLAB File Exchange. 28 Nov 2014. Web.

Bird, Richard. “A Simple, Solar Spectral Model for Direct-Normal and Diffuse Horizontal Irradiance.” *Solar Energy.* 32:4:461-471. 1985.

Brine, D.T. and Iqbal, M.. “Diffuse and Global Solar Spectral Irradiance Under Cloudless Skies.” *Solar Energy.* 30:5:447-453. 1983.

Patel, Hiraj and Agarwal, Vivek. “MATLAB-Based Modeling to Study the Effects of Partial Shading on PV Array Characteristics.” *IEEE Transaction on Energy Conversion.* 23:1. March 2008.

Sparrow, E.M. Eckert, E.R.G, and Jonsson, V.K.. “An Enclosure Theory for Radiative Exchange Between Specularly and Diffusely Reflecting Surfaces.” *Journal of Heat Transfer*. 84:4:294-299. 1 November 1992.

Leica ScanStation P40 3D Laser Scanner, (accessed on 12/18/15), http://www.fltgeosystems.com

Fritz Kasten and Andrew T. Young, "Revised optical air mass tables and approximation formula," Appl. Opt. 28, 4735-4738 (1989).

Nguyen, Xuan Hieu. “MATLAB/Simulink Based Modeling to Study Effect of Partial Shadow on Solar Photovoltaic Array.” Environmental Systems Research, 4:20 (2015).

Villalva, Gazoli, and Filho. “Comprehensive Approach to Modeling and Simulation of Photovoltaic Arrays,” IEEE Transactions on Power Electronics, 24:5 (2009).

Amit Yadav and S.S. Chandel. “Tilt Angle Optimization to Maximize Incident Solar Radiation: A Review,” Renewable and Sustainable Energy Reviews, 23 (2013).

Dick Jordan and Sarah Kurtz. “Photovoltaic Degradation Rates - An Analytical Review,” NREL, 5200-51664 (2012).

Hofierka, Jaroslav and Kanuk, Jan. “Assessment of Photovoltaic Potential in Urban Areas Using Open-Source Solar Radiation Tools.” Renewable Energy. 34. 2009.

Renewable Resource Data Center (RReDC). *Solar Resource Information.* NREL. *http://www.nrel.gov/rredc/solar_resource.html Accessed 2016.*

"PCD file format tutorial - Documentation - Point Cloud Library (PCL)." 2011. 10 May. 2016 <http://pointclouds.org/documentation/tutorials/pcd_file_format.php>

"Measurement of Solar Transmittance through Plate Glass : SHIMADZU ..." 2011. 10 May. 2016 <http://www.shimadzu.com/an/industry/ceramicsmetalsmining/chem0501005.htm>

VIII. Acknowledgements

We would like to thank these people

for helping with the project and offering

their ideas and support.

Prof. Sam Keene

Prof. Toby Cumberbatch

Prof. Joseph Cataldo

Dolen Le

David Shechtman