SOLAR PANEL HOTSPOT DETECTION: USING DRONES (UAVS) TO DETECT MALFUNCTIONS IN SOLAR ARRAYS

MAYLEEN CORTEZ

Contents

1. Introduction 2
2. Background and Motivation 2
3. Project Roadmap 5
3.1. Environment Set-Up 5
3.2. Image Thresholding 6
3.3. Contours 8
3.4. Panorama Stitching 10
4. Results and Discussion 11
4.1. Using Contours to Detect Hotspots 12
4.2. Contour Features 14
4.3. Making One Image Out of Two 14
5. Future Work 15
6. Conclusion 17
References 18

Date: December 17, 2018.

1. Introduction

In this paper I will discuss a project I have been working on during the Fall 2018 semester at California State University Channel Islands as part of an undergraduate mathematics research course. My project has been to develop a collection of python scripts that will allow for real-time detection of hotspots on solar panels via drone technology. In the following sections, I will discuss the background and motivation for this project, elaborate on the process I have taken in planning and approaching the problem, reflect on my work and discuss paths forward.

2. Background and Motivation

In this section, I introduce solar energy and its relevance, as well as the use for solar panels and their maintenance. The sections ends with a discussion about the origins of the project for the Naval Facilities and Expeditionary Warfare Command at Port Hueneme, CA and how I came to be involved in it.

In 2009 the American Recovery and Reinvestment Act (ARRA) was passed in response to the Great Recession. ARRA, also known as the Stimulus Bill, had several components and directed funds to provide relief to families, modernize government infrastructure and many other causes. Through ARRA, more than $31 billion funds were also allocated for clean renewable energy projects [10]. As a result, numerous solar arrays were installed worldwide at United States Navy installations, but no budget was provided for maintaining them. For almost 10 years, these solar arrays have been, from a maintenance point of view, left untouched [5].

Solar panels are made from very durable material and are designed to

withstand diverse weather conditions. In general, they have a lifespan of up to 25 years [6]. Since solar panels have no moving parts, one of the main focuses of maintenance is maintaining efficiency by keeping the panels clear of dust, foliage, snow and other debris. Ground-mounted solar arrays are closer to dust and pollen than roof-mounted arrays so, cleaning requirements are greater. There is also the issue of overgrown vegetation, which means that mowing, trimming, tree removal and other similar tasks may be included in solar array maintenance. On the other hand, roof-mounted arrays can be difficult to inspect, clean and more costly to maintain, depending on the layout of the roof. Maintaining roof-mounted arrays is much more dangerous and requires different safety equipment, as well as other training. The steeper and more complex the roof is, the more safety requirements are added to maintenance [3]. It can be expensive to send out technicians to inspect these solar arrays for defects and malfunctions because of the safety precautions that must be taken and the risks involved. Sometimes the danger presented by maintaining these roof-mounted solar panels outweighs the benefit of sending up a technician, especially when it is not certain that there is a solar panel malfunction anyway [5].

Luke Richards, an Aerial Targets Engineer for the US Navy, along with some colleagues, came up with the idea that drone technology could be used to mitigate the costs and risks involved in solar panel maintenance. Solar panels are a way to harness solar energy, one of the cheapest and most abundant renewable energy sources available to us today. Using renewable energy instead of fossil fuels reduces the amount of greenhouse gasses we emit into the environment. We can prevent environmental damages and lessen health risks by reducing the

emittance of these pollutants into the atmosphere and in turn, reducing air pollution. Solar energy is a clean and free energy source that produces no waste and to harness it, we use solar panels [11]. With a plethora of unmaintained solar arrays available across naval installations worldwide, Luke and his colleagues recognized a potential for increased efficiency, decreased spending and a step towards environmental health.

In 2018, Dr. Jason Miller, a mathematics professor at California State University Channel Islands, was invited to participate in an Office of Naval Research Summer Faculty Research Program to work with Mr. Richards in exploring the way computer vision technology would aid in the task of drone inspections of solar arrays. That summer, Dr. Miller gathered digital images of solar panels using infrared cameras mounted on a small quadcopter drone. He also got to work on writing some code in the popular programming language Python that would allow him to process the data gathered and come up with a way to effectively use drone technology to inspect solar arrays. Although he made reasonable progress in those few months, by the time fall semester came around, there was still plenty of work. In the fall, Dr. Miller decided to extend the offer of working on this project to the group of students in an undergraduate mathematics research course he was teaching. I took the course and decided to tackle the problem together with another student. Since the data we will be using to analyze solar arrays is in the form of videos and images, we have had to use digital image processing techniques. My main focus has been in the detection of "hotspots" on solar arrays using thermal images, since these are generally indicative of solar panel failure [1].

3. Project Roadmap

The ultimate goal of this project is to create an interface that allows for real-time hotspot detection via an unmanned aerial system (UAS). To accomplish this, I decided it would be best to create a set of smaller, more attainable goals. Throughout the Fall 2018 semester, my primary goal has been to create a collection of scripts that allows for automated detection of hotspots on solar panels. Since the UAS takes video data while flying over the solar arrays, we figure that to detect hotspots in real-time, we may need to be able to stitch together several frames of a video and create one cohesive image of the entire solar array, keeping in mind that some may be up to several thousand yards long [1]. For this reason, I made panorama stitching a "subgoal." I have used the data gathered by Dr. Miller over the summer and the skills I've acquired through my undergraduate mathematics and computer science courses to take some of the work others have done in digital image processing (through the Python library OpenCV [2]) and apply them to this project. In this section I will describe the approaches I took to the project. I emphasize this section contains details of my *process*, not my results. The results are described in Section 4.

3.1. **Environment Set-Up.** A major aspect of this project has revolved around programming in Python, in particular using the NumPy [**?**] and OpenCV [2] libraries. NumPy is a library that allows us to work with and manipulate multi-dimensional arrays. This is useful because images are interpreted as multi-dimensional arrays by computers. The nature of our goal falls into the category of "computer vision," a branch of research that deals with developing mathematical techniques that allow for a computer to interpret image data with the same depth,

detail and accuracy as a human being [9]. OpenCV is a Python library that contains computer vision tools. The majority of the coding and data sharing for this project has been completed using CoCalc, an online platform that allows for researchers, students and instructors to collaborate on projects using a variety of programming languages and development tools [1] [8].

3.2. **Image Thresholding.** When I first began my research, I focused on using greyscale infrared images of solar panels for simplicity. One thing I noticed was that the hotspots on solar panels were much darker than the rest of the solar panel. I briefly mentioned in Section 3.1 that images are treated as multi-dimensional arrays by computers. Greyscale images are two-dimensional arrays whose entries are real-valued and range from 0 (black) to 255 (white). Thresholding is a technique that isolates particular groups of pixels in an image, or components in an array, that exceed a certain value, or intensity. Since the hotspots are closer in intensity (from a human visual perspective) to black than white, it made sense to use thresholding to detect the hotspots. There are six thresholding methods already built into OpenCV, so I worked with the tools and techniques already available. After some trial and error with sample images, I was able to come up with an algorithm[2] that could detect hotspots on a cropped image

[1]On CoCalc, it is useful to use Jupyter notebooks to test and run the code, as they allow for images to be displayed. For more information about CoCalc, refer to its documentation, found at `https://doc.cocalc.com/`. To see the code I have written for this project, you can refer to `https://tinyurl.com/yd4j267e`

[2]Please note that the "imshow" method that comes with cv2 does not work inside the Jupyter shell notebook on CoCalc. To display images, the library matplotlib must be imported and its particular "imshow" method should be used to display images.

with reasonable success. Here, cropped image refers to an image of a solar panel that has been cropped to eliminate as much background as possible. The code returns the location [3] of each pixel that meets the thresholding criteria.

Next, I decided to work on improving the code to draw an outline around the found hotspots, give them unique labels and retrieve some useful information, such as their center and area. For this, I turned to another tool from OpenCV called blob detection. Though I was able to use the blob detection technique to draw circles on the hotspots, I had a lot of trouble getting any useful information from them because the circles did not outline the blobs with high precision on some hotspots, as show in Figure 1. As I was reading up on segmentation methods in

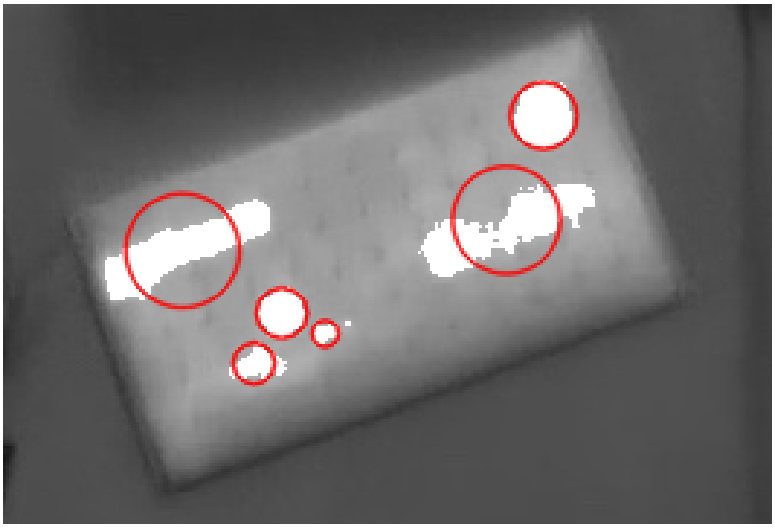


FIGURE 1. Circles drawn on solar panel hotspots by blob detection algorithm.

[3] array index

OpenCV, I stumbled onto contours. As I read more of the documentation for contours in OpenCV, it became evident that they might offer a better solution for hotspot detection.

3.3. **Contours.** The contours library in OpenCV proved to be very useful in building a collection of scripts to detect hotspots on solar panels. With this new tool, I modified my project goals by creating a new series of smaller goals. My approach was, in some sense, to "start from scratch." Although I had learned many basics during the thresholding phase, I did not want to treat "simple" tasks, like displaying an image, as trivial[4]. So, I created a new Jupyter Python shell on CoCalc to start my work with contours and my first goal was to read an image into the program correctly. The next step was to just find the contours in an image[5].

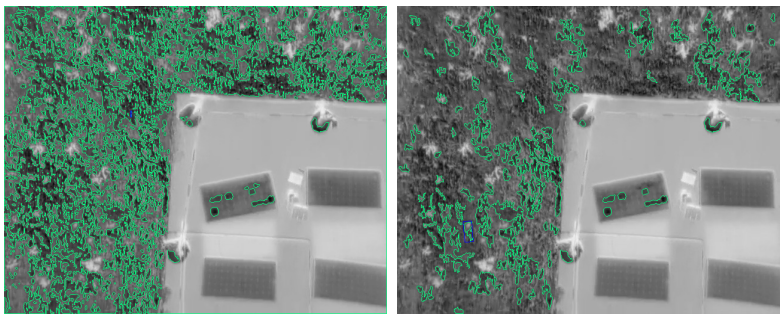
After successfully finding contours, I focused my attention on the information the contours function in OpenCV can return beyond the graphical representation on the contours themselves. For example, online I found sample code for calculating information about a particular contour such as the moment, area, perimeter and bounding rectangle[6] [2]. After I was able to working code working for finding the moments, areas, perimeters and bounding boxes, I began testing my code using cropped images from the data gathered by Dr. Miller over the summer.

[4]I think this allowed me to avoid complacency and carelessness when writing code.

[5]Note that much of the code I used to achieve these smaller goals came from the contours documentation and the tutorials found on `https://opencv-python-tutroals.readthedocs.io/en/latest/`. Most of the errors I encountered during this stage were fixed by reading the documentation or searching the error up on Google.

[6]For a complete list, please explore the link in the previous footnote.

With minor tweaking and adjusting, I was able to identify the hotspots in several cropped images, as well as outline them in color and show their bounding rectangle. Once I began testing with uncropped images, I found that although the algorithm still picked the hotspots on the solar panels, it also picked up numerous contours in the background, namely the vegetation surrounding the area of the solar panels (see Figure 2a). I reduced the number of "extra" contours in the images by iterating through their areas, eliminating any that were under a certain value. This helped somewhat, but the script still picks up too many contours (see Figure 2b). Hence, we have good and working code for



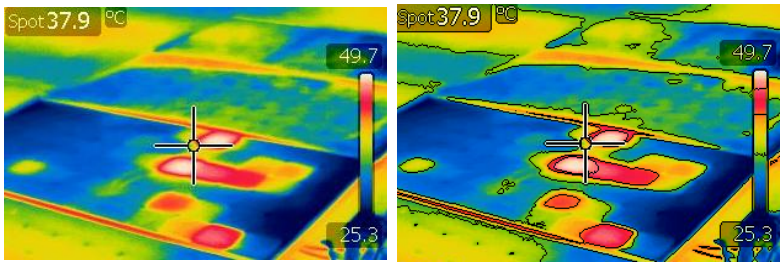
(A) Frame 45, contours identified.

(B) Frame 45, contours identified with altered script.

FIGURE 2. Shown here are two of the exact same image, one the result of a contour detecting script with no modification to eliminate small contours (a) , and the other the result of a program that ignores contours of an area smaller than a specified amount (b).

images where there is hardly any vegetation showing, but have not been able to modify the algorithm enough to recognize when the contour is a hotspot versus vegetation.

Up until this point, I was only working with greyscale infrared images. Since I also had some color infrared images of the solar panels available, I wanted to adjust the code as necessary[7] to correctly identify the hotspots in these images. As shown in Figure 3, the adjusted code [8] is able to identify hotspots on colored infrared images but, as with the greyscale images, picks up too many contours. Since I made what



(A) Color infrared image of solar panel.

(B) Contours detected on colored infrared image.

FIGURE 3. Here we show the results of the contours script on a colored infared image of a solar panel. On the left is the original image. On the right, the same image is shown but with contours outlined in black by the code. The code identifies any pixels with values we perceive as red, white and yellow as hotspots.

I felt was reasonable progress towards detecting hotspots on a solar panel, I decided to move on for a bit and focus on panorama stitching.

3.4. **Panorama Stitching.** I was not able to spend much time with this part of the project, but I did create a subgoal: to develop code that

[7]As it turns out, there were only a few lines that needed to be modified.

[8]Note that in the GitHub repository, this code is in a different script than the code used to identify hotspots on greyscale images.

allowed a user to give two "ideal" images and, in return, save a stitched together, or panorama, image. To do this, I used code[9] already developed and debugged by Dr. Adrian Rosebrock[10] [7]. However, because of his advanced knowledge of Python and OpenCV, Dr. Rosebrock's algorithm included some classes and methods that had to be modified to work on CoCalc[11] More details on the results of this algorithm can be found in Section 4.3.

Throughout this section, I discussed the approaches I took to develop a program for real-time detection of hotspots on solar panels via an unmanned aerial system (UAS). By breaking this goal down into smaller goals and working through the associated tasks, I attained reasonable progress. Outlined in Section 4 are the details of the results of these efforts and my work for the Fall 2018 semester.

4. Results and Discussion

This project began with an ambitious goal: to create a program or interface that allows unmanned aerial systems (UAS) to detect hotspots in real-time on solar arrays placed in United States Navy installations.

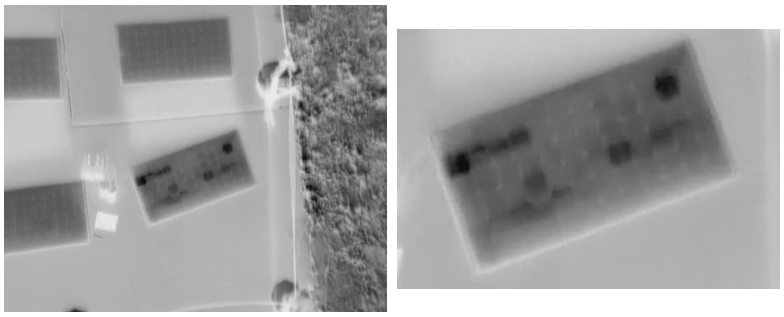
[9]To see this code refer to `https://www.pyimagesearch.com/2016/01/11/opencv-panorama-stitching/`

[10]author of the book "Deep Learning for Computer Vision with Python" and creator of the website `https://www.pyimagesearch.com`

[11]These changes were minor. Many of his scripts require the use of a command line or terminal to give the program its arguments. On CoCalc, using a terminal is possible but images cannot be displayed. To display images, a Jupyter Notebook must be used. However, I could not find a way to access a terminal for the shell. Hence, I needed to adjust the code to either avoid using the terminal, or avoid the need for displaying the images. You can find this code at `https://tinyurl.com/yd4j267e`.

I have focused on developing a collection of Python scripts to post-process still images, or individual video frames, and detect the hotspots on solar arrays. The results of these efforts and this process will be outlined in the following subsections.

4.1. **Using Contours to Detect Hotspots.** I had the most success in locating hotspots on solar panels when I used the contours function and its related methods in the OpenCV library. As mentioned in Section 3, I began using these techniques and testing Python scripts using cropped images from the data Dr. Miller acquired during the summer of 2018. I have included an example of one of the images and the resulting cropped image in Figure 4.



(A) Frame 22. (B) Frame 22 Cropped.

FIGURE 4. On the left (a) is a sample unmodified image. On the right (b) is the same image, cropped to eliminate as much background as possible and focus on one solar panel.

Running the contours script[12] shows successful detection of the hotspots

[12]To view the code used to obtain Figure 5, refer to the following link: `https://tinyurl.com/yd4j267e`. More examples of images and the image resulting from running it through this script are also available at the same link.

on the solar panel in Figure 4b. The contours, which outline the regions that are hotspots, are outlined with a bright green border.

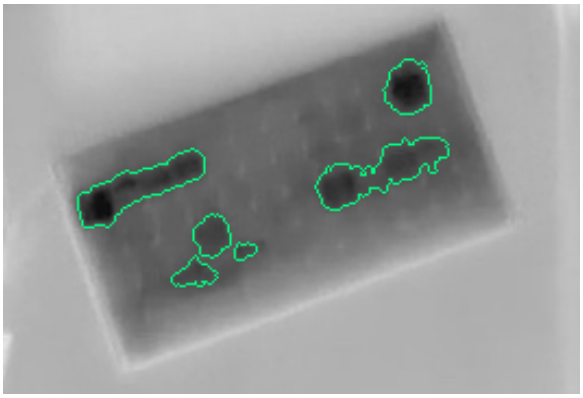


FIGURE 5. Frame 22, hotspots identified.

Although this script works relatively well for cropped images, it is less accurate for the original images. Hotspots are still detected, but the program picks up many other regions of comparable temperature. The vegetation in the background and the bottom of the cones around the pavement are identified by the contour algorithm. This is a problem because our objective is to detect hotspots on solar panels, not any object with warmer temperatures. Figure 2a shows an example of this problem. The script was modified to ignore any contours below a certain size, but still picks up too many. The result of this altered script[13] is shown with a sample image in Figure 2b. Besides identifying the hotspots in a solar panel, we also want to return useful information about the hotspot, such as its area or intensity.

[13]To see the modified script, please refer to the following link: `https://tinyurl.com/yd4j267e`. More examples of images and the image resulting from running it through this script are also available at the same link.

4.2. **Contour Features.** While there is enough information in the OpenCV documentation to identify many features for each contour, I focused on obtaining each contour's moment (with respect to the center of mass), area, perimeter and bounding rectangle. In each case, the coordinates and quantities returned are in terms of pixels. Since solar panel technicians are going to need information about the real-life solar panel (rather than just an image), we will need to convert the information returned by these contours functions into useful information. This is further discussed in Section 5.

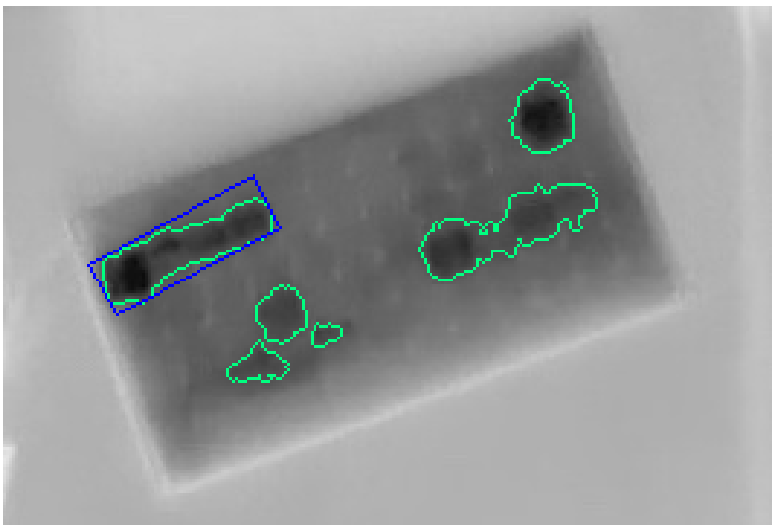


FIGURE 6. Example of a rotated bounding rectangle shown on one of the contours.

4.3. **Making One Image Out of Two.** Using a script[14] created by Dr. Adrian Rosebrock, a computer scientist and computer vision researcher [7], I successfully stitched together two images. For this algorithm to work on two images, there need to be enough pixels in each

[14]For a "CoCalc Friendly" version of the script, see `https://tinyurl.com/yd4j267e`.

image that match the other. In other words, this algorithm does not take two completely separate and distinct images and stitch them together, side by side. It relies on there being some overlap between the two to stitch them together. The algorithm also works left to right, meaning that if you want to stitch together images obtaining from panning over something in an up/down motion, the algorithm would need to be modified. To better visualize this, you can think of the panorama feature for an iPhone camera, where an individual "pans" their phone slowly from left to right to get a wider image of something. An outline of the next steps to take in this particular aspect of the project, *i.e.* the panorama stitching aspect, is included in Section 5.

In this section, I discussed the results of my research and work for the Fall 2018 semester. This included the results of using OpenCV's contour library to detect hotspots and return information about them. This also included using a image-stitching algorithm to take two images with some overlap and create one, cohesive image from the two. The following section will discuss future work and improvements.

5. Future Work

This section outlines the next steps an individual could take to make further progress in this project.

As mentioned in Section 4.1, the contour finding script for detecting hotspots on solar panels picks up extra regions when the image is not cropped. Identifying these "irrelevant" regions that are not solar panel hotspots does not help us achieve our goal and makes it so that our script returns more information than is useful. To continue this project, a good next step would be in brainstorming ways and implementing

some ideas to alter the script so that only useful information is retured. Some ideas are to

- include an algorithm that isolates the solar panels in the image frame in the script for contour detection and restricts the areas of the image where the contour algorithm looks for contours.
- determine (or find) a probabilistic model that can be included in the script so that contours with "low probability" of being on a solar panel are ignored.

In Section 4.2, we described some of the characteristics about the contours that we can return via the contours function in OpenCV, such as perimeter, bounding rectangle, area, etc. However, the information we return is related to the image being an array of pixels, not a real-life object in three-dimensional space. For example, the area returned for each contour is with respect to pixels, not feet or meters. Hence, to further advance and return information that a Navy technician could use, we might need to develop some the following:

- a way to extract information about the exact location and possible dimensions of the solar panels as the UAS flies over the solar array
- an algorithm that can estimate the real-life dimensions of the solar panel based on 2-D images
- an algorithm that takes speculations of solar panels and uses them to turn the pixel data into real-life data

As mentioned in Section 4.3, there are certain restrictions on the panorama stitching algorithm. First, the algorithm only works when you feed it images in a left to right order. If you feed the rightmost image first, the algorithm cannot detect key points correctly and will not return a

stitched image. If the script is kept this way, a drone would be restricted to moving across solar arrays in a left-to-right motion always, which is impractical, or the images will have to be flipped afterwards and then fed into the script. The latter is a viable option, however maybe not the most efficient. A better idea may be to determine a modification that allows the script to be more flexible. This could mean the ability to feed images in right-to-left, up-to-down, and down-to-up. The algorithm also only stitches two images at a time. We either need to modify the script to stitch multiple images at a time, or include some sort of loop to build a large panoramic image two images at a time.

6. Conclusion

When I began this project, I was hoping to create an elaborate and efficient program for the Naval Facilities and Expeditionary Warfare Command at Port Hueneme, CA to use in their inspection of solar panels. I knew that time was limited and that I might not be able to create a finished "product," but I thought I might get somewhat close. The grand objective was to create an interface that allows for real-time detection of hotspots on solar arrays via unmanned aerial systems. To better my chances, I broke the ultimate goal into smaller targets that I needed to hit in order to reach the final mark. Using OpenCV, I was able to detect hotspots on solar panels and return information about them. I also used a script created by Dr. Rosebrock [7] to begin skimming the surface of image stitching. The results I achieved were helpful in getting closer to a final product, but there is still a lot of work that needs to be done to finish this project.

References

[1] Miller, Jason. *Personal Communication,* 9-17-2018.

[2] Mordvintsev, Alexander, *OpenCV - Python Tutorials,* 2013, Retrieved from https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html

[3] National Renewable Energy Laboratory, *Best Practices in Photovoltaic System Operations and Maintenance,* 2016, Retrieved from https://www.solarfinancecouncil.org/resources/Best-Practices-in-PV-O&M.pdf.

[4] NumPy developers, *Numpy*, 2018, Retrieved from http://www.numpy.org/.

[5] Richards, Luke. *Personal Communication,* 12-05-2018.

[6] Richardson, Luke, *How long will my solar panels last?,* 2018, Retrieved from https://news.energysage.com/how-long-do-solar-panels-last/.

[7] Rosebrock, Adrian, *OpenCV Tutorials, Resources and Guides*, 2018, Retrieved from https://www.pyimagesearch.com/opencv-tutorials-resources-guides/.

[8] Snyder, Hal, *SageMathCloud is Now CoCalc,* 2017, Retrieved from http://blog.sagemath.com/cocalc/2017/05/20/smc-is-now-cocalc.html.

[9] Szeliski, Richard, *Computer Vision: Algorithms and Applications,* 2010, Retrieved from http://szeliski.org/Book/.

[10] US Department of Energy, *Successes of the Recovery Act,* 2012, Retrieved from https://www.energy.gov/sites/prod/files/RecoveryActSuccess_Jan2012final.pdf.

[11] Wiser, Ryan, Trieu Mai, Dev Millstein, Jordan Macknick, Alberta Carpenter, Stuart Cohen, Wesley Cole, Bethany Frew, and Garvin A. Heath, *On the Path to SunShot: The Environmental and Public Health Benefits of Achieving High Penetrations of Solar Energy in the United States,* Golden, CO: National Renewable Energy Laboratory, Retrieved from https://www.nrel.gov/docs/fy16osti/65628.pdf.

1 University Dr, Camarillo, CA

Email address: mayleen.cortez136@@csuci.edu