B.L.I.N.D.S

Blocking Light IN Domestic Spaces

Jeff Chen, Elizabeth Chuei, Dianne Ge

Department of Electrical and Computer Engineering,
Carnegie Mellon University

***Abstract*—Sunlight often serves as an annoyance to people by obstructing their vision, and adversely affecting their visual comfort, eye health, and productivity. However, completely eliminating natural light has been shown to negatively impact a person's mood and mental health. To combat this issue, BLINDS will eliminate the manual process of turning blinds while maintaining optimal room's lighting. The system will automatically adjust blinds to appropriate height if it detects sunlight hitting a person in the room.**

***Index Terms*—LIDAR, Smart Blinds, Computer Vision, OpenCV, Arduino, Photoresistor, Suncalc, GeoPy**

I. Introduction

Previous research shows that direct exposure to sunlight affects people negatively in a multitude of ways, including obstructing their vision and negatively impacting their eyesight [1]. Overexposure to the sun's harmful UV rays can also cause a variety of eye diseases [2]. However, completely removing sunlight is not an optimal solution, because eliminating natural light can prove detrimental to a person's mood, mental health, and productivity [3]. In other words, to maximize quality of life, one must continuously adjust a room's blinds in a way that minimizes the amount of light that hits one's face, but maximizes the amount of natural light in the room. However, continuously adjusting blinds can disrupt concentration and in turn, decrease one's productivity. To combat this, we aim to create blinds, for people who spend a lot of time at home, that will adjust automatically in a way that prevents light from hitting a person's face, but lets in maximal sunlight. This type of product is particularly relevant, as the number of people who work remotely and spend copious amounts of time at home has skyrocketed since the COVID-19 pandemic [4].

Although other motorized blinds exist, most require the use of a remote control that the user has to press to adjust the blinds. However, pressing a remote control also disrupts concentration, which conflicts with its original purpose of providing convenience and allowing the user to focus on other tasks. Other competing technologies include blinds that move according to the time the sun rises and sets. However, this type of system fails to take into account the position of the user in the room, and whether the light hits them at that location. Our proposed solution seeks to increase a person's quality of life by balancing the amount of sunlight they are exposed to, while solving the existing issues of alternate solutions.

II. Use-Case Requirements

We want our system to be beneficial enough so that a user may be able to focus on the task at hand without being hindered by sunlight. In order to quantify this, our group ran a short survey on a group of college students to ask what the minimum accuracy rate they would be able to tolerate for purchasing a system like this would be. The average came out to 90%. Given this answer, we aim to have our overall system be able to block sunlight from a person's face, by adjusting the blinds so that the light projected from the window into the room will only reach a person's face 90% of the time.

On the other hand, we do not want our blinds to be a source of distraction either. Many widely used face detection systems take under 1 second to detect a face, which is quite fast [5]. However, a constantly moving blinds system may be more of a nuisance than a source of convenience. It may also potentially frighten users and create an unpleasant experience. In the same survey as mentioned above, we asked participants whether they would prefer a constantly moving blinds system, or one that would wait for a set amount of time for a user to stop moving before adjusting the blinds. Additionally, if a participant answered that they preferred the system to wait, we asked how long of a period the system should standby before making the adjustment. The results showed that the participants would be startled if the blinds were constantly moving, and on average preferred a wait time of 10 seconds. Therefore, we want the feedback latency, which is the time it will take for our blinds to react after a user's movement, to be around 10 seconds.

We also want our physical latency (the time it takes for the blinds to physically adjust upwards or downwards) to be on par with existing motorized blinds. Popular motorized blinds on the market take around 60 seconds to fully roll upwards or downwards [6]. Therefore, our motorized blinds system should have at most a 60 second physical latency in order to compete with similar existing products.

Finally, we want to be able to achieve our accuracy goal within a reasonable workspace: the average bedroom size in the US, which is 132 ft^2 [7]. This means we want to be able to block sunlight from a user's face within our accuracy goal in a 132 ft^2 space.

III. Architecture and/or Principle of Operation

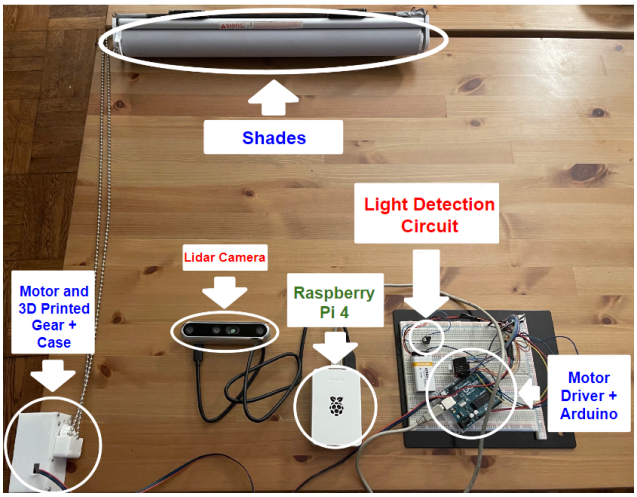


Fig. 1 Full physical blinds system

Our full system consists of 3 primary interconnected systems, each with separate subsystems:

1. Sensor System
    a. LIDAR camera
    b. Light Detection Circuit
2. Processing Unit (RPi)
    a. User Position Algorithm
    b. Sun Calculation API
    c. Light Area of Effect Algorithm (LAOE)
3. Motor Control

The sensor system (1) consists of two "sensors", each for different purposes. The first sensor is the Intel RealSense Depth Camera D455. This integrated depth and camera system will provide information to determine the 3D coordinates of where a person is located within the room in question. Another sensor is the sunlight detection circuit, which uses photoresistors to determine if the weather is sunny enough to warrant adjusting the blinds.

The Processing Unit, the RPi, (2) will be where the necessary calculations are done on the software side. One of the software systems is a user position calculation algorithm. This employs OpenCV and depth information from the LIDAR to calculate the 3D coordinates of a user in the room space. Another component of the software system is the sun calculation API. This function will use the location of the window (latitude, longitude) in addition to the time of day to find the azimuth and altitude of the sun relative to the blinds system. Finally, the last algorithm we will be writing is the LAOE algorithm. Using the measurements of the window, the user position, and the sun data, this algorithm will determine an adjustment so that sunlight is not in the user's face, as well as maximizing the amount of sunlight entering the room.

The motor control, or the physical blinds system (3), will consist of a few working parts. This consists of an Arduino Uno, which takes in adjustments as inputs, and the motor system attached to the controls of the blind, which will receive the adjustments from the Arduino. The motor system will be attached through a 3-D printed gear that will hold tightly onto the beads of the blinds' cord lock.

The relationship between these parts and a greater breakdown of each subsystem are illustrated and can be better seen in Appendix C at the end of this document.

One change we have made to this system since the design report is the use of the magnetometer. We have opted to remove that from the system and instead use a one-time input at the beginning of installing the system at a new location.

IV. Design Requirements

We want our sensor subsystem to contain parts that are at least 90% accurate, to match with our overall accuracy requirements. To limit the impact of polling rate on feedback latency, we want our sensor system to poll for real-world information as much as possible. However, because feedback latency is mainly limited by processing unit limitations, which will take at least a second, it's best to wait for the processing unit to perform its calculations before polling again. So we want our system to poll for information at least once per second.

As for the photoresistor circuit in this system, the threshold that determines whether light is considered sunlight, which determines whether the blinds move or not, will be determined in the future with thorough testing with real sunlight. Because this threshold is dependent on a variety of factors, including the type of photoresistor used, it has not yet been determined.

Our motor control system needs to be able to fully open/close the blinds in 60 seconds. Because the standard window height is 60 inches, the motors in our system need to be able to rotate at a fast enough speed to allow the blinds to move at a rate of 1 inch per second. The motor system must also be able to allow for precise position control by increments of 1 cm with a margin of 0.5cm. We currently believe that a 1 cm precision is precise enough to cut off light at the optimal level, but this is subject to change, as it depends on the results of the previously mentioned survey involving user satisfaction in maximizing the sunlight in the room.

To meet our overall accuracy requirement of 90% accuracy in a 132 ft^2 room, we want our processing unit to correctly identify whether the user's position is in the area of direct sunlight identified by the LAOE algorithm 90% of the time. Although we know that the total accuracy a typical user would want is 90%, we do not know how much error is allowed within each subsystem to achieve this overall error of 90%. We do know that if a single subsystem is less than 90% accurate then the overall accuracy of our integrated system can't reach 90%. Thus we are currently giving each subsystem a hard 90% accuracy minimum.

To be able to capture more real-world data and to increase the reaction speed to the user's movements, we also want our processing and calculations to take less than 1 second; this will help us more closely reach a feedback latency of 10 seconds. And, once again, although the requirements for minimizing sunlight on the user's face for this are fairly defined, more requirements will be made later on after the

survey about the blinds' position to maximize sunlight in the room has been conducted.

V. Design Trade Studies

A. User Position Extraction

There were other methods we were considering using to calculate the position of the user in the room. One such method was using multiple cameras around the room, and using triangulation to identify the position of the user in the room. This method would allow us to forgo looking for a LIDAR device, and would be fairly accurate. However, using multiple cameras complicates the design, as the particular angle and position of which the cameras are put up have to be taken into account. It would also provide inconvenience to the user when setting up our BLINDS. To slightly simplify the design, as well as provide more convenience to the user, we decided to use a LIDAR and camera integrated device. Another similar method that was under consideration was using Bluetooth beacons, but that requires the user to hold onto a device to communicate with, which is also inconvenient for the user, which contradicts our goal to maximize convenience for the user.

Another design choice we made was the choice of using Haar cascades over other forms of facial detection. Although Haar cascades tend to be slightly less accurate compared to other methods, it is also faster than other methods. This is also particularly pertinent because its lower latency makes it not only run well on a microcontroller, but helps us achieve our latency requirements. Yet another design choice we made was using a Fast Super-Resolution Convolutional Neural Network (FSRCNN) model to increase the resolution of the camera image, over other models. We decided to use a FSRCNN model over an EDSR model because although EDSR's gives the best results in terms of accuracy, it is slower [10]. Because we want our BLINDS to react in real-time, it is better to use a model like FSRCNN.

B. Sun Position Calculation

Initially, we considered a few different options for calculating the data on the sun. One was the use of a sun sensor or a sundial to determine where the sun is in respect to the window. The benefit of these physical forms of data measurement is that it can help us determine whether or not sunlight is being blocked from the window in some way, such as by weather or by objects such as trees or buildings. For instance, a lighter shadow or lack of shadow for the sundial may indicate that it is cloudy. However, there are quite a few drawbacks. The sun sensor would be an expensive option for finding information that could be calculated with just location and time of day. There would also be the issue of where exactly such a device would be placed, as putting it outside would make it separate from the entire blinds system. Our team also considered the use of a sundial, but methods of retrieving data and data accuracy would also be quite messy. We needed something that would be as accurate as possible, so we decided to simply calculate the location of the sun using Python modules, which would also only require the time, date and location.

C. Sensor System

To support the software solution we chose in the previous section, the sensors we need are a photoresistor, a Depth Camera, and a Microcontroller with analog pins. For our photoresistor, we chose the Elagoo photoresistor, as this is a component that we already own and will work sufficiently for our purposes. Finally, we chose the Arduino Uno for our microcontroller because this was also a device that we already had our hands on, and a component we have confidence and experience with in the past.

For our depth camera, we were originally contemplating between the Astra Depth Camera from Orbbec 3D, Intel RealSense Depth Camera L515, and RealSense Depth Camera D455. Our original plan was to use the Intel RealSense Depth Camera L515 because the CMU ECE inventory conveniently already owned one, and its range of up to 10 meters would fit our design requirements. Unfortunately, after some initial testing, we learned that the L515's effective range is reduced down to 1.5 meters in sunlight, which is a huge problem given the nature of our project. We then considered the Astra Depth Camera from Orbbec 3D, as it was relatively cheap and had a range up to 8 meters. However, we decided against it because there were no reviews for this depth camera, and therefore we were not able to confirm if the range was reliable. The Astra also did not have good documentation which we feel would be a huge issue. Thus, we settled with the more expensive camera, the Intel RealSense Depth Camera D455, which has many reviews affirming its 4.2 meter range even in direct sunlight and is well-documented enough for us to quickly learn how to utilize.

D. Motor Control

For our motorized blinds, we were choosing between the Automatic Window Roller Blinds [11] and the Arduino Automatic Blinds Opener [12]. The Automatic Window Roller Blinds is a design plan to make our own custom blinds, where the stepper motor will directly turn the rollers the blind's cloth is attached to. On the other hand, the Arduino Automatic Blinds Opener is a design which has an external motor with a gear that hooks onto the beaded string of the shade and controls the shade through the string. Both designs take 60 seconds to fully unroll/roll up. We decided to go with the Blinds Opener over the Roller Blinds because the Blinds Opener seems more versatile as it works with every blind as long as it has beaded strings. The Blinds Opener also fits our project better because the motor can sit on the window sill; this way, we don't need a long wire from our RPi all the way to the top of the window, which would have been necessary if

we were to use the other design.

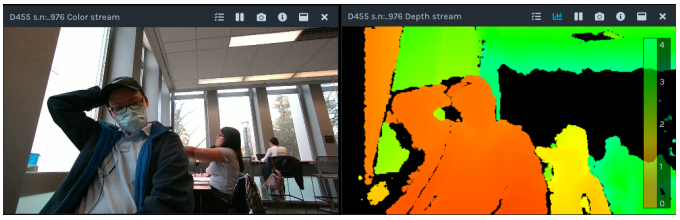


Figure 1: Sample Output of the Intel Realsense Depth Camera D455

VI. System Implementation

A. *Sensor System*

The sensor subsystem consists of the Intel Realsense Depth Camera D455, and light detection circuit. The Intel Realsense Depth Camera D455 provides both a RGB camera feed and also lidar data at a resolution of 1240px x 720 px at 30fps as shown below in figure x. The Camera would be connected to the RPi via a USB-C to USB 3.1 cable. The lidar camera worked as expected. The sample output of this device can be seen in Fig. 1.

The other component of the sensor system is the light detection circuit. The light detection circuit is constructed by connecting a 10 kΩ resistor in series with a photoresistor to form a voltage divider. The voltage reading would be connected to the Arduino Uno analog pin with a wire. The

Arduino Uno would measure the voltage level across the photoresistor and determine if it's below the threshold to be considered direct sunlight. The result would send light detection results to the RPi serially using PySerial via a USB connection when the RPi sends a request for the data.

One of the sensors we decided to no longer utilize was the magnetometer. This was because we realized that it doesn't make sense to have a whole sensor for data we will only collect once. We determined that making the installer manually enter the window orientation into the BLINDS once is worth the $5 dollar decrease in the cost of the BLINDS considering that the installer already has to input the user's address anyways.

B. *Processing Unit - User Position Extraction*

From the sensor system, the processing unit (the RPi) obtains real-world information. This information includes, but is not limited to, an image of the user and the room, as well as data from the LIDAR, which depicts the distance of the system to objects in the room, as shown in the previous section.

Using this information, the User Position Extraction Algorithm is run on the processing unit. Its goal is to obtain the location of the user's chin, which will be used in tandem with the LAOE algorithm to determine if the user is hit by direct light or not. First, it uses OpenCV's Haar cascades to perform face detection on the camera input which it will obtain using the pyrealsense2 API, to detect the location of the user's face. Both the frontal and side profile cascades will be used, so that we can identify the user's face in multiple orientations. We can convert the face detection output to pixels on the user's chin from the camera input. Given the pixels of the user's chin, we can map this pixel to pixel to the LIDAR data using the pyrealsense2 API, and from that we can identify the radial distance the user's chin is from the system. Then, by using the number of pixels away the user is from the center of the image, we can identify the angle of the user to the system. Using the distance and angle of the user, we can determine the 3-D coordinates of the user, as represented by the coordinates of their chin.

In our initial design, we wanted to use image processing to increase image resolution using OpenCV's Super Resolution module and TensorFlow's Fast Super-Resolution Convolutional Neural Network (FSRCNN) model. However, this was deemed unnecessary, as upon further testing, OpenCV's face detection worked adequately for the distances described in our use case.

Although the final method to calculate User Position was consistent with our initial design document, another calculation method was also attempted. This method accounted for the non-linearity in the relationship between the number of pixels away from the center of the camera's view and the angle away from the center [15]. However, after the testing of this method, it was shown that although using this method decreased error in the x-direction, it increased error in the y-direction, and this created an averagely larger error across all directions, so we reverted to the previous method mentioned above.

C. *Processing Unit - LAOE Algorithm*

As shown in the full architecture diagram in Appendix C and described in Section III of this document, the LAOE algorithm receives data on the user's position, the sun (azimuth, altitude, photoresistors), and the window's orientation. The user position is found from the algorithm above, the sun data is found through the python modules Suncalc.py and Geopy.py and the photoresistor circuit., and the window orientation is set as a one time input before the use of the product. We want to check first whether or not the weather is affecting the sunlight. This is done through the data sent by the photoresistor sensor circuit. If there is no direct sunlight, we will keep the windows open/open them fully as the sunlight will not bother anyone. Otherwise, in order to find the adjustment that needs to be made to the blinds, our algorithm will be divided into a few functions. First, it will figure out if the user's position lies within the "area of effect" of the sun. This area, or more accurately, volume, is a trapezoidal prism of the projection of the sun from the window into the room. The idea is rooted in the fact that light travels in straight rays from the sun [8]. The rays will hit the corners of the window and travel into the room, at certain angles of altitude and azimuth. The projection of light traversing into the room at each of the four corners is what forms this box-shaped prism. It is better visualized and illustrated in Fig 2. In order to do this, there are three helper functions.

The first function is the getProjectionCoords() function,

which maps the four coordinates of the window into the projection of the same four coordinates onto the floor by using the azimuth and altitude of the sun as the directions of the rays as they enter the room. Because the sun is very far away from the window in question, we can estimate all rays entering the room to have the same altitude and azimuth. The function will return the four coordinates of the corners of the light cast by the window. The coordinates are represented with the x-value representing the distance to the left or right of the window, the y-value representing the distance from the wall where the window is located, and the z-value being the physical height within the room. The origin is set at the floor under the window, with x value at the middle of the window.

The second function, intersect, determines whether or not there is an intersection between the user and the LAOE. Using the eight corners we have found so far (four from the corners of the window + four from the corners of light projected onto the floor), we now have the eight vertices of the trapezoidal prism formed from the light. We can simplify the intersection to a calculation of whether or not the point lies between the two lines existing in each plane (XY plane and YZ plane). The equations for the four lines (two for each plane) can be found from the two coordinate points we have for each line (the previously calculated eight total points). This is illustrated in Fig. 3 and 4, and the intersection is calculated with a simple inequality intersection calculation.

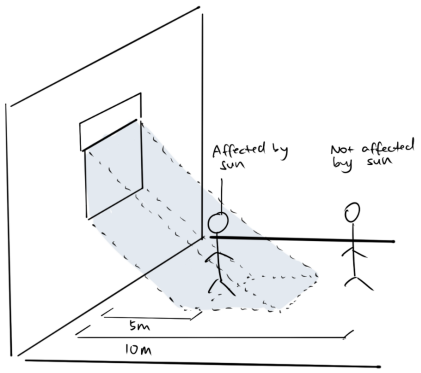


Figure 2: Visualization of Light Area of Effect

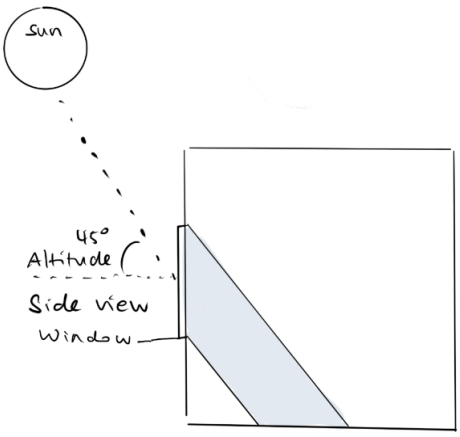


Figure 3: View from the YZ plane of LAOE

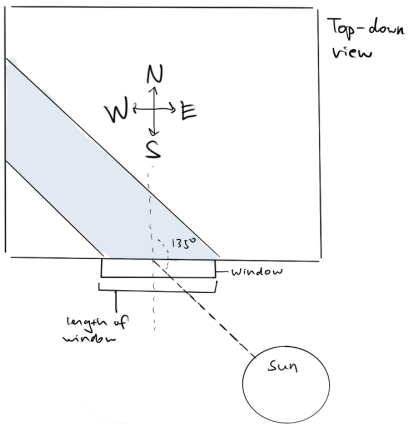


Figure 4: View from XY plane of LAOE

If we did not find an intersection, we can open the blinds fully so that light is maximized. However, if we did find an intersection, we must adjust the blinds. The last function, blindsChange(), calculates the adjustment that must be made in order to prevent light from hitting the user's face. To do this, our algorithm will backwards trace a ray hitting a target point (the bottom of the person's face) back until it reaches the wall where the window is located. This ray will use the same altitude and azimuth for its direction. This will allow us to calculate what the z-value of the bottom of the blinds should be (bottom of the blinds = top of the window), and adjust the blinds accordingly to that position. The result is sent to the motor control.

There were a few bugs throughout the process of writing this code. Some prominent issues were flipped intersections and leniency for intersections. The flipped intersections happened as the sun moved from one side of the window to the other, and to mitigate this, we simply checked for whether or not a person's coordinates were between the maximum and minimum of the inequalities. Additionally, we also wanted greater leniency for the edge cases of the intersection function, as it is better to close the blinds in the case that a person is on the edge rather than not close the blinds, and we set an leniency of 1 cm within the bounds of the edges of the LAOE.

D. Motor Control

This subsystem has 3 components: Usongshine Nema 17 Stepper Motor 42BYGH, HiLetgo 5pcs A4988 Stepstick Stepper Motor Driver Module, and Arduino Uno Rev3. The process begins with the RPi sending a request to set the Blind to a certain position via a USB-to-USB connection serially. The Arduino Uno would then send the series of stepper motor control signals (Step size, direction, enable, etc) to the Stepper Motor Driver Module which interfaces with the Stepper Motor itself to rotate the number of rotations in units of 1/16 step to achieve the requested position. The motor would be connected to a gear that hooks onto the beads on the blinds string to control the blinds. This whole architecture is shown in figure 5 on page 6 and the physical system is labeled in figure 2 on page 2.

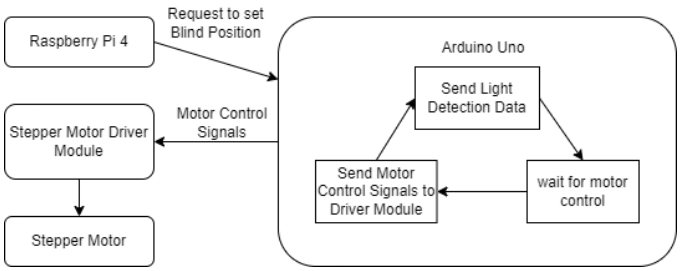


Figure 5: Motor Control System Architecture

VII. Test, Verification and Validation

A. Results for Sensor System

We tested the Intel RealSense Depth Camera D455 by having a person stand at a wide variety of angles (the whole FOV of the camera) and distances (from 0.5m - 6m) in front of the camera. We visually confirmed that the person is portrayed at the right spot on the camera and measured the actual distance between the person and the camera. The D455 datasheet stated to expect ~5% and our test matched that.

We also tested the light detection circuit by moving it between the circuit between areas of direct sunlight and areas of ambient light or shadow. After some number of iterations to find the best threshold value, our circuit had an 100% accuracy in determining whether it's exposed to direct sunlight.

B. Results for User Position Extraction

To test the User Position Extraction, we made a person stand at set distances (1 meter, 2 meters, and 3 meters) away from the system, and we tested each distance both at an angle of 30 degrees to the right of the camera, and 30 degrees to the left of the camera. Then, we ran our User Position Extraction algorithm and compared its outputs to the actual measured angle and distances of the user. We want to achieve two different types of goals, and thus will have two different quantitative targets, when testing: Maximizing sunlight in the room, and minimizing the sunlight that hits the user's face. Although we initially tested these distances at an angle of zero degrees (at the center of the camera), we decided not to as this skewed our data a lot. This is because we wanted to use percent error as our metric, since our system has to be more precise for people closer to the window who are more likely to be hit by the sun. However, this meant that the x-distance error would be massive for degrees near zero, even if the actual difference was only a couple of decimeters. Other measures of error were looked into to address this issue, but none fit our needs as accurately as percent error, so we decided to just test at certain angles.

For minimizing sunlight, we wanted to meet our overall accuracy requirement of 90% accuracy in a 132 ft2 space; this accuracy was defined as the percentage of times our system accurately identifies the user is in the area of the room that gets hit by direct sunlight, as found by the LAOE algorithm. To test this 90% accuracy, the User Position Extraction algorithm was broken down into four, separately tested components: Determining where the chin of the user is using OpenCV, using that information to find the radial distance of the user using LIDAR data, calculating the angle of the user using pixels to the center of the image, and overall percent error of our calculated position compared to the actual user position. For the first one, we defined our accuracy as the percentage of times that the found location of the chin, given an image, was at or within 5 cm of the actual chin 90% of the time, within a 5 cm range, to ensure light does not hit above the chin. During testing, we were fairly accurate in this aspect and met our initial accuracy goals, as this was largely dependent on OpenCV.

The second portion, finding the radial distance of the user using LIDAR data, is solely determined by how accurate the LIDAR data is. We aimed for this distance to be within ~5% of the actual distance; this is more heavily discussed in the sensor system section. For the third portion, we wanted the angle of the user to be as precise as possible, down to the pixel, as this can heavily affect whether the user is determined to be standing in the light or not, and affect our accuracy goals. Using the Depth Camera D455's maximum resolution and angle of view, we calculated each pixel to be approximately 0.703 degrees horizontally and 0.090 degrees vertically, so we initially wanted our calculated angle to be within those angle ranges [13]. Although we later realized that at extreme degrees, our system is not as accurate as the relationship between angle and pixel are not linearly related, other calculations attempting to address this issue surprisingly increased our error, so we decided to stick with our initial linear interpretation that we came up with. Overall, we met our goal, using our linear assumption. For the last portion, the overall accuracy, we had 87% for the x-direction, 85% for y-direction, and 94% for the z-direction, with an overall average of all the directions at 88.7%. Although this did not meet our goal of 90%, the difference is fairly negligible, and it's likely our system still functions fairly well.

C. Results for LAOE Algorithm

In order to test the correctness of the LAOE program, we tested the coordinate values of the rectangular projection from the sun through the window and onto the floor. We collected data points at varying times of the day with sunlight, from 10 am to 6 pm. These data points are the coordinates points of the rectangular prism projected onto the ground, with four points per data point representing the top left, top right, bottom left, and bottom right corners of the rectangle. There were a total of 32 data points collected, varying in time, location, and corner of the projection.

The accuracy was measured as relative accuracy, and the results over 32 data points had 96.93% accuracy. This met and exceeded our requirement of 90% accuracy.

Some possible limiting factors of this accuracy include human error in measurements, issues with the window orientation, and small differences in the altitude and azimuth of a specific location. The greatest issue with window

orientation was that our measurement device was not giving a consistent number. It gave values within a 20 degree range, which is a huge discrepancy, and it led us to results that were very different from the expected value. However, once we had gotten a consistently correct orientation measurement, the results were very close to the expected values, with within 10% error.

D. *Results for Motor Control*

There are two factors we care about for the motor: physical latency and consistency. Physical latency is the duration it takes the motor to fully unroll or to fully roll back up. Our user requirements stated that we needed the physical latency to be under 60 seconds. We tested this by making the shade to fully unroll and record that value then repeat this process while the shade rolls back up. After repeating this process a few times and adjusting the motor speed we got our physical latency to 58.6s. Increase in motor speed causes inconsistencies where the beads will occasionally pop out of the gear causing the motor to not move the intended distance by a few milli-metter. This issue could be combated by placing the motor in a specific orientation in respect to the beaded string to reduce motor jitter's effect on blind consistency. With optimal orientation, the consistency of the motor is ~98%.

E. *Results for Overall Integrated System*

Overall testing for our project entails testing whether or not a change is made and whether or not that change is correct. We conducted tests including variations and edge cases such as when a person is on the edge of the LAOE, not within the LAOE, and around the inside of the LAOE. Overall, we got a result of 90% success, where success is denoted by the existence of a change and the correct change. This is exactly the value in our system requirements. Some limitations include limitations of the subsystems, which is further explained in their individual subsections.

VIII. Project Management

A. *Schedule*

Our project is on schedule the whole project, and made no adjustments. The Gantt chart for our schedule is shown below in Appendix B.

B. *Team Member Responsibilities*

All the group members have rough experience in all the subsystems of the project, so we will all have a secondary task of helping each other with subtasks. However to utilize the specializations of each group member, the project work has been divided into two main parts: The hardware component, which involves working with the Arduino and motors, and the software component, which largely consists of implementing the LAOE and User Position Extraction algorithm on the Raspberry Pi. The work has been split up such that Jeff is in charge of the hardware portion of the project, and Dianne and Elizabeth will work closely together on the software components. All group members will be responsible for integration of the different components of the project.

C. *Bill of Materials and Budget*

A breakdown of the materials bought and used can be found in Appendix A below. The majority of our planned purchases were bought and used except for the magnetometer. There are also a bunch of PVC pipes and LED lights that were bought since we realized we needed them to build a fake window frame for demo purposes.

D. *Risk Management*

Our project was a success and was able to stay on schedule because of the risk mitigation and management plans we utilized. The risk mitigation plans we employed are rigorous early research and planning phase, flexible work schedule, strong communication within the group, and leaving buffers for the unexpected.

The first risk management plan and arguably the most impactful risk mitigation factor was our early research and planning phase. We performed thorough research on all the possible solution options for the first two weeks after deciding on the problem we wanted to tackle. This allowed us to make a realistic schedule (Appendix B) and a good solution design that was mostly untouched.

Another reason why we think we were able to successfully avoid major risks was the flexible work schedule and strong emphasis on team communication within the group. We recognized early on that having a tight schedule with lots of deadlines for smaller tasks will only hinder us because we all have other school commitments we have to address and breaking down tasks too specifically would only serve as a limiting factor for our adaptability. We also recognized that the downside of a looser schedule is that people could end up cramming their portion last minute and not make it in time. We offsetted this by maintaining constant communication between group members to hold each other accountable.

The last risk mitigation plan we had in place was leaving buffers in case of unforeseen circumstances. This buffer is primarily in two forms, a time buffer and a monetary buffer. We recognized that our product could only be tested on sunny days so we set a 4 week period for integration and testing to maximize our chance of getting enough sunny days for field testing. We also set a monetary buffer of about $50 in case we needed anything that was unexpected during our planning which we used later to purchase stuff needed for our demo.

IX. Ethical Issues

Decisions about technology design and engineering can influence how people communicate, work, travel, and live, and it is therefore important to consider ethical issues that pertain to how our product is designed. One such ethical concern is the existence of bias in the facial detection

algorithm we use in our product to identify the position of the user. Because of how data is collected, datasets are weak to bias, and tend not to be equally representative of every race and gender. Research shows that for Haar cascades and their given default datasets (which we used in our project), the most detectable class of faces is European men [14]. In other words, our product will have a higher rate of failure for those who aren't European men. Although this is a non-negligible issue, creating our own dataset to train on would likely result in worse bias, as it is likely we would not be able to collect a dataset large enough to generalize better than the default datasets.

One other ethical issue is the security of our device. Although we initially did not plan to allow remote access to our device, we found that this made it impossible to set up our device without a monitor. Each time our device was disconnected from power, a monitor was required to re-run our program on the Raspberry Pi. To resolve this issue, we set up SSH on our Raspberry Pi. Although this increases the attack vectors for our device, we thought the ease in set-up was well worth the trade-off. Moreover, the Raspberry Pi is password protected, so although this places the burden of security on the user (to pick a good password), the Pi is still fairly secure.

Another issue is that we determine what is the best amount of sunlight for the user to adjust our blinds. We assumed that the ideal amount of light would be minimizing light on the face, but maximizing overall light in the room. Although we may have satisfied most users, in reality, the best amount of light in reality is fairly user-dependent. If a user is very deficient in vitamin D, they may want some sunlight on their face. On the other extreme, if a user is very sensitive to sunlight, they may want sunlight blocked from their entire body. Another case where the user might want the blinds closed more than they will be automatically adjusted is if they are doing something private in their home, and want the blinds completely closed. Our product does not take into account these situations, and we recommend that future work on our project involves creating a web application where the user can override our optimal light algorithm if they so wish.

X. Related Work

Serena Shades by Lutron [9] are the only smart blinds on the market that have a light optimization feature built into the blinds. The blinds, over the course of the day, automatically tilt themselves at an angle that prevents direct sunlight from entering the house, but allows natural light to fill the room. The Natural Light Optimization algorithm that Serena Shades implements utilizes a person's address and window orientation, information which has to be manually inputted via the Lutron app, to calculate the angle the blinds should be tilted at.

Our project differs from Serena Shades by a few key factors. For one, when calculating how our blinds should move in order to optimize light in a room, our project not only takes in a person's address and window orientation, but it also takes into account a user's position within a room. This means that our BLINDS allows more natural light in a room, by letting in direct sunlight that does not negatively impact or hit the user. Additionally, our BLINDS also considers if light is really coming through the window or not, as a factor of whether or not an adjustment should be made, while Serena Shades functions the same regardless even if the weather doesn't have sunlight. Our system will only adjust if it detects the presence of direct sunlight through our light sensing circuit, thereby expanding on Serena Shades' Natural Light Optimization algorithm.

XI. Summary

Our system was able to meet most of the design specifications. In some cases, such as the LAOE accuracy and feedback latency, and range of position detection, our project was able to greatly exceed our goals. The specifications that fell short were some of the accuracy measurements along certain axes in user position detection. This was limited in part to the accuracy of the face detection model.

Overall, this project was an incredible learning experience. We gained so much knowledge on topics we had not come across before, such as working with LIDAR, computer vision, and hands-on project pipeline experience. If future groups were to take on a project similar to this, we would recommend keeping track of upcoming sunny days to test data, as well as making sure the physical system has an easy way to be transported and displayed.

Glossary of Acronyms

API - Application Programming Interface
LAOE - Light Area Of Effect Algorithm
RPi – Raspberry Pi

References

[1] Boyce, Peter R. "Review: The Impact of Light in Buildings on Human Health." Indoor and Built Environment, vol. 19, no. 1, Feb. 2010, pp. 8–20, 10.1177/1420326x09358028.

[2] Young RW. The family of sunlight-related eye diseases. Optometry and Vision Science : Official Publication of the American Academy of Optometry. 1994 Feb;71(2):125-144. DOI: 10.1097/00006324-199402000-00013. PMID: 8152745.

[3] Shishegar, N, and M Boubekri. "Natural Light and Productivity: Analyzing the Impacts of Daylighting on Students' and Workers' Health and Alertness." International Journal of Advances in Chemical Engineering and Biological Sciences, vol. 3, no. 1, 21 May 2016, 10.15242/ijacebs.ae0416104.

[4] Mitchell, Travis. "Covid-19 Pandemic Continues to Reshape Work in America." Pew Research Center's Social & Demographic Trends Project, Pew Research Center, 23 Mar. 2022, https://www.pewresearch.org/social-trends/2022/02/16/covid-19-pandemic-continues-to-reshape-work-in-america/.

[5] Kazanskiy, Nikolay, et al. 'Performance Analysis of Real-Time Face Detection System Based on Stream Data Mining Frameworks'. Procedia Engineering, vol. 201, 2017, pp. 806–816, https://doi.org10.1016/j.proeng.2017.09.602.

[6] Graywind, "Graywind Motorized Blackout Roller Shades." Graywind, 2020,

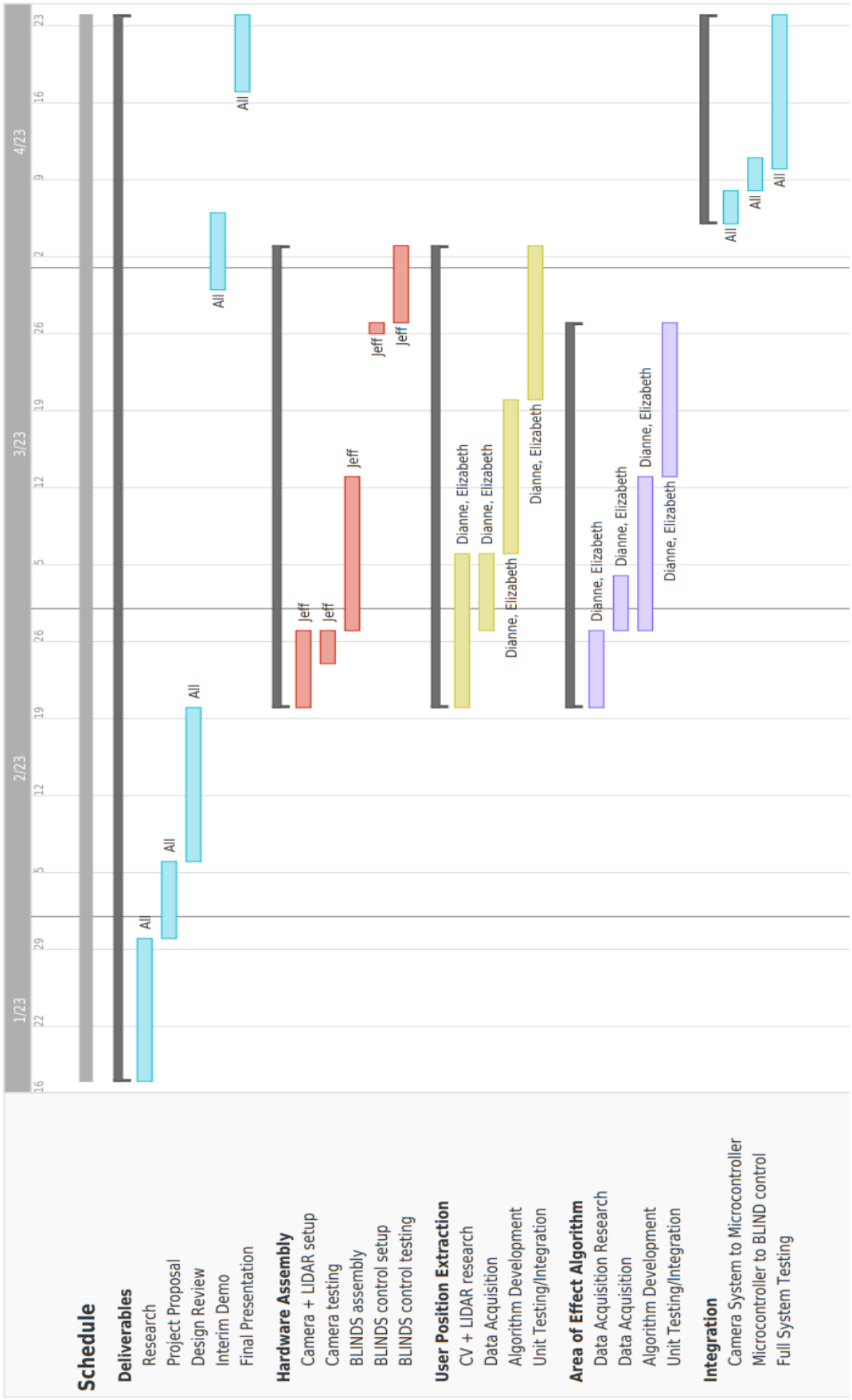
https://www.graywindblinds.com/collections/roller-shades/products/smart-shades-blackout-designed
[7] Helling, Author Andrew. “Average Bedroom Sizes in a Home - The Ultimate Guide.” REthority, 14 Feb. 2023, https://rethority.com/average-bedroom-size/.
[8] Bird, J. O., and P. J. Chivers. ‘31 - Light Rays’. Newnes Engineering and Physical Science Pocket Book, edited by J. O. Bird and P. J. Chivers, Newnes, 1993, pp. 248–255, https://doi.org10.1016/B978-0-7506-1683-6.50034-5.
[9] Serena Shades. SerenaShades, Lutron, 2016, https://www.serenashades.com/.
[10] Vardan Agarwal, Lipi Patnaik, et al. “Super Resolution in Opencv.” LearnOpenCV, 18 Oct. 2021, https://learnopencv.com/super-resolution-in-opencv/#sec3.
[11] “Automatic Window Roller Blinds.” Projecthub.arduino.cc, 10 June 2021, projecthub.arduino.cc/twinsen01/85418824-2918-44ac-87c5-dc7deb4be4b5.
[12] Klements, Michael. “Arduino Automatic Blind Opener .” The DIY Life, 2 Apr. 2020, www.the-diy-life.com/arduino-automatic-blind-opener-works-with-a-remote-control-alexa/.
[13] Coffin, Jerry. “OpenCV: Calculate Angle between Camera and Pixel.” Stack Overflow, 1 May 1960, https://stackoverflow.com/questions/17499409/opencv-calculate-angle-between-camera-and-pixel.
[14] Rudinskaya, E. (2020). Face detection accuracy study based on race and gender factor using Haar cascades.
[15] TravisJ. “Convert a Pixel Displacement to Angular Rotation?” Mathematics Stack Exchange, 10 June 2015, https://math.stackexchange.com/questions/1320285/convert-a-pixel-displacement-to-angular-rotation.

Appendix A: Bill of Materials and Budget

Color Code: Blue - Planned and bought
Green - Planned and not bought
Red- Not Planned and bought

Description	Model	Manufacturer	Source	Quantity	Cost
Lidar and Camera Integrated System	RealSense Depth Camera D455	Intel	Intel	1	$453.27
Window Blinds	Grey - 20" W x 72" H	Homebox	Amazon	1	$31.99
Screws kit	810Pcs M3 x 4/6/8/10/12/14/16/18/20 mm Screw Assortment Kit	VIGRUE	Amazon	1	$17.99
Stepper Motor Driver Module	A4988	HiLetgo	Amazon	1	$10.19
Stepper Motor	42BYGH	Usongshine	Amazon	1	$10.99
Main Computer	Raspberry Pi 4 8 GB	Raspberry Pi	ECE Receiving	1	$0
Motor Controller	Arduino Uno Rev3	Arduino	N/A	1	$0
Photoresistors	N/A	Elegoo	N/A	4	$0
Magnetometer	MMC5603	Adafruit	Adafruit	1	$5.95
3D Printed Pieces	N/A	N/A	TechSpark	1	$17.66
Battery	9v Cell Battery	Amazon	N/A	1	$0
PVC Pipes	3/4 in. x 10 ft PVC	Charlotte Pipe	Home Depot	2	$12.58
PVC Elbow	3/4 in. S x S Elbow Fitting	Charlotte Pipe	Home Depot	6	$4.74
PVC Tee	3/4 in. S x S x S Tee	Charlotte Pipe	Home Depot	2	$1.64
LED Light	TCB001	UBeesize	Amazon	1	$32.60
					$593.65

Appendix B: Gantt Chart of Schedule



Appendix C: Full Architecture Block Diagram