Camerazzi

Author: Mimi Niou, Cornelia Chow, Adriel Mendoza:

Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—** This report details the conception and design of an autonomous robotic cameraman intended for social gatherings and/or professional events. Hiring a photographer often involves unavailability, unreliability, intrusiveness, bias, and even latency in delivery of the photos.

Our solution offers an automated robot that's always available, reliable, nonintrusive, and unbiased. In addition, it ensures ample margins around faces and delivers photos instantly. Our robot does this efficiently with software algorithms, requiring lower power and less storage space than other implementations such as one that deletes subject-less photos after an event, demonstrating the capabilities and advancements of electrical and computer engineering.

***Index Terms*—** autonomous, camera, design, face detection, iRobot, OpenCV, robot, Raspberry Pi, Roomba, sensors

I. Introduction

In this day and age, recording social and professional events through photographs is a common occurrence. However, cameramen often have busy schedules and sometimes are not only intrusive but also biased when taking photos. Our project was inspired by a problem presented to us by the Robotics Club at CMU. This club organizes several events throughout the year and they always have trouble booking a photographer because of the challenges detailed in the abstract. It is for these reasons that student volunteers are typically sought out. As the number of volunteers has been steadily declining, we came up with the solution of designing a robot that could take on the duties of a photographer.

Camerazzi aims to be a nonintrusive, unbiased, available, and reliable alternative to the typical cameraman that would be booked for an event. Our project provides a solution particularly suitable for recurring events due to reduction in costs and the constant availability. While other implementations like taking numerous snapshots randomly at an event are also viable, our robot uses the integration of software and hardware to efficiently take photos of the important subjects, the people, at an event. This way, we reduce the power consumption required, save storage space, and eliminates the need for post-analysis. To achieve this, we want to always have photos to deliver to the user. Our robot must take at least 1 photo per person in the room per hour if there are less than 20 people in the room. If there are more than 20 people in the room, there is a minimum requirement of 20 photos per hour no matter how many people there are. The details of these goals will be outlined below.

II. Design Requirements

In order to measure success for our robot and the pictures that it takes, we have defined the following requirements.

As an overarching requirement, we expect our robot to capture a minimum number of photos, with a given room size, number of people in the room, speed of the Roomba, and duration of time the Roomba is active. By researching standing crowd density [7] we have set an ideal crowd density of 1/4 people per square meter. This will allow enough room for our robot to roam and capture photos at ideal distances.

Therefore, given l, the length of the room in meters, w, the width of the room in meters, n, the number of people in the room, and t, the duration of the event in hours, the minimum number of photos we expect is calculated using the following equations and testing constraints. We assume here that it will take approximately 3 minutes to find a face, adjust robot position, and capture a photo, so in an hour, 20 shots is the maximum number of photos that could be taken.

$$a,\ area\ of\ room\ =\ l\ \cdot w$$

$$0\ <\ n\ \leq\ (\frac{1\ person}{4\ square\ meters})\ \cdot\ a$$

$$t \leq 3.5\ hours\ (battery\ life\ of\ iRobot\ Create\ Robot)$$

$$minimum\ \#\ photos\ =\ (\ min(n,\ t * 20)\ *\ t\ )$$

In other words, we want to have at least as many photos as attendees in the room taken every hour. This requirement is imperative to the success of our project, as the main function of our robot is the ability to take photos at an event, so photos captured act as a key success factor of our project.

Our first low-level requirement is that the robot must be at least 3 feet away from humans in front of it and behind it. We set this requirement because academic papers such as Mumm & Mutlu's *Human-Robot Proxemics* [6] describe a comfortable distance between robots and humans as being approximately 3 feet. This also helps us achieve the nonintrusive aspect we wanted for our robot. We will evaluate this requirement by laying out a measuring tape in front a stationary person, setting the robot to move towards the person, and checking to see how far the robot stops from the person.

Another requirement we've established is that robot stopping

latency must be less than 1 second. We set this as our maximum latency because we want our robot to move at a speed of .5 feet per second. In our software, we will be detecting when the robot is 3.5 feet away from a human, and so a latency of 1 second will account for the extra .5 feet between the distance we want to stop as dictated by the software and the distance we want to stop based on our requirements. We will evaluate this requirement by recording the time that a stop command is sent to the robot, and the time that the robot actually stops.

The next requirement we've set is that the robot must be able to detect 90% of human faces in real time. We've set it to this value because we have found that using this algorithm on still images typically achieves a 95% success rate. But since we're using this algorithm on continuous video, we've slightly lowered the success rate to 90%. We will evaluate this measurement by manually counting the number of faces our algorithm detects at points in time during the demo, and dividing it by the number of faces that were actually present in the frames.

Another requirement we've set is that 100% of the photos taken must include a human. We have set this requirement because we don't want our robot to take extraneous photos when it should be spending its time taking useful photos. This is consistent with our decision to use this implementation, as opposed to one where take many photos and delete the ones that are not usable. We will evaluate this requirement by examining every picture taken at the demo and confirming that there is a human in the photo.

Our next requirement is that every photo must have a 5% margin between the borders of the face and the borders of the image. This is to ensure that no heads get cut off when the picture is taken. We will evaluate this requirement similarly to the previous requirement by examining every picture taken at the demo and confirming that there is a 5% margin between the face borders and the image borders.

The next requirement is that the width of each face in the image must be at least 10% of the width of the image. We came up with this figure because after examining many photos, a face width that is 10% of the image is characteristic of a decent picture. Any less than this would mean that the person's face is out of focus in the shot, and it will appear as though we have taken a random picture. This requirement, in addition to our margin requirement, contributes toward the consistency aspect we wanted for our project. We will evaluate this requirement by examining every picture taken at the demo and confirming that the face width is at least 10% the width of the image.

Another requirement we've established is that 100% of the photos taken should be sent to the Google Drive folder over Wi-Fi. We've set this requirement to ensure that our users received all the photos that were taken. This also helps us achieve the instant access to photos aspect we wanted for our robot. We will evaluate this requirement by recording the number of photos taken at the demo, and comparing it to the number of photos sent to the Google Drive folder.

III. Architecture and/or Principle of Operation

For our mobile robot, the iRobot Create Programmable Robot will act as the moving base, upon which additional hardware components will be mounted.

Our software-related system architecture revolves around the use of a Raspberry Pi, which will act as the computer for the entirety of our software components. Hosted on the Raspberry Pi, will be our motion control, image optimization, and collision detection software. Specifically, we will be using the PyCreate2 library to program the movement of the Roomba. For photo optimization, we will be using OpenCV datasets to detect faces and determine when and where to capture photos. Lastly, we will be using SciPy to process and interpolate the thermal camera sensor and IR sensor data for detecting collisions. In addition, we will be using three Arduino Uno's to simply retrieve the thermal camera sensor and IR sensor data, and relay the data to the raspberry pi.

For photo capture, we will be using the Raspberry Pi Camera Module, which is easily integrated onto the Raspberry Pi and is able to capture 8MP photos.

For collision detection, we will be using 2 thermal camera sensors and 18 IR sensors to detect when people/objects are in close proximity, so that our robot can stop moving and avoid collisions. We will use bicubic interpolation through SciPy to construct greater datasets and expand the limited output from the thermal camera sensors.

For further control over image capabilities, we plan to use a motorized stick which will move up or down as specified by our image optimization algorithm. We will integrate the motorized stick into our system using a Bluetooth connection with our raspberry pi.

Lastly, we will be using an LCD screen, connected to the raspberry pi, to display simple prompts to users to indicate when a photo will be captured.

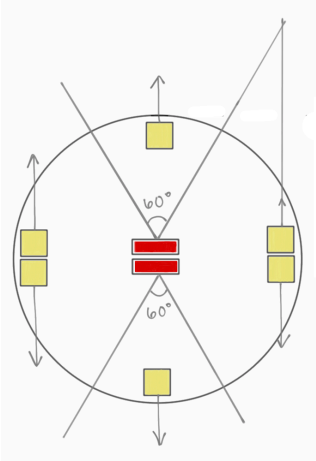
Our block diagram (Fig. 1) demonstrates all of the above described components, as well as how they will communicate and how information will flow in our system.

IV. Design Trade Studies

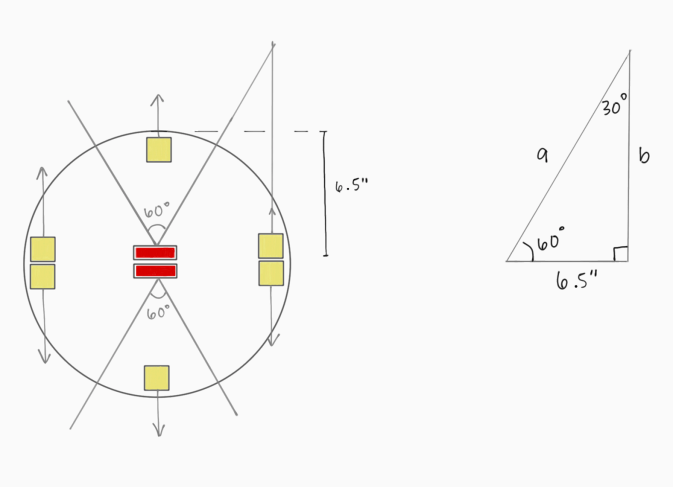
We decided to use the Sharp GP2Y0A02YK0F IR proximity sensor, because it is better at reliably providing data between a range of .5 feet to 5 feet. Since we are detecting for a distance of 3.5 feet, this sensor shows a clear advantage over other IR proximity sensors such as the UNCL4010 proximity sensor, which can only accurately detect collisions up to 7.5 inches away. We also found that this sensor's ability to bounce reflective light off the clothes of a human make it better than ultrasonic sensors. We wanted there to be 3 IR proximity sensors tacked on top of each for each required position on the Roomba. Using 3 sensors allows us to eliminate outlier data that comes from a sensor that occasionally sends bad data. Additionally, we wanted for there to be 6 positions on the robot for the IR sensors. 3 facing front on the left, center, and right; and 3 facing back on the left, center and right. The robot will only be moving either forward or backwards, so we reasoned that we would only need to account for those 2 directions. The left, center, and right placements are meant to account for collisions that would happen either directly in front of/behind the robot, or at the edges of the robot. So with the 3 sensors per position, and 6 position on the robot, we need to purchase 18 IR proximity sensors.

In addition to the IR proximity sensor, we will be using the Adafruit AMG8833 8x8 Thermal Camera Sensor. Our initial reason for using this sensor was because it was able to detect heat signatures a far distance away (ie. up to 23 feet) and it had a fairly decent field of view (ie. 60°). We were going to use this sensor to detect humans so that our robot can maintain the 3 feet requirement of buffer space. We quickly realized that this sensor could not accurately determine depth, so we plan on using it conjunction with our IR proximity sensor. Thus, for our collision detection mechanism, we need the IR proximity to detect for collisions, and we need the thermal camera sensor to determine whether that collision is a human or an object. We decided that we only need two thermal camera sensors (one in front, and one in back) because with the 60° field of view, we can capture what is seen by the 9 IR proximity sensors that face the same direction.

This diagram describes how the sensors will be placed on the robot.



The distance between the IR proximity sensor and the intersection of the thermal camera sensor's field of view and the IR proximity sensor's field of view must be less 42.5". This is to ensure that the thermal camera will also detect anything detected by the left and right IR proximity sensors. Since we know the angle between the right IR proximity sensor and the right side of the thermal camera's field of vision is 60°, and that distance between the IR sensor and the thermal camera sensor is 6.5", we can determine that the intersection point is 11.2583", which is less than 42.5". Thus we can conclude, that we only need one thermal camera sensor per direction.



a = 13"
b = 11.2583"

We decided to use 3 Arduino UNOs for some of the signal processing. This is because we wanted to free up the Raspberry Pi's computational power for image processing which needs to be as fast as possible. The Arduino UNO has 6 pins for transmitting analog data, and since we are using 18 IR proximity sensors for collision detection, we need 3 Arduinos.

We decided to use the Raspberry Pi 3 Model B+, because it is Wi-Fi and Bluetooth capability. By having Wi-Fi capability, we are able to easily debug our code by simply powering the Raspberry Pi on campus and being on the CMU-DEVICE network. By having Bluetooth capability, we can pair our motorized stick with our Raspberry Pi, in order to move the camera up and down. This model also has a CSI port which allows us to seamlessly integrate our Camera module. Lastly, this model has the most USB ports (ie. 4), which works perfectly because we will need 4 ports to connect to the 3 Arduinos and the Roomba.

We decided to use the Raspberry Pi Camera Module V2 because of its ability to take 8 MP pictures, which we deemed sufficient for our purpose.

V. System Description

A. Flow

Although there are many different scenarios we have considered to be potential contexts for our robot, the behavior will follow a general pattern of roaming, sensing for humans, adjusting, and capturing a photo (Fig. 2). More specifically, Camerazzi will begin roaming in a straight path until it either senses a human or collides with a wall or other object, using its thermal and IR sensors. In the case that Camerazzi senses a human, it will begin using OpenCV and the raspberry pi camera to scan for faces in view. If faces are not detected within the frame, the robot will continue roaming.

If one or more faces are detected, our algorithm will run a series of calculations to determine if the photo should be taken and how the robot should adjust its position and camera height. The details of this algorithm are detailed in Section B where we describe our photo optimization subsystem. Before photo capture, we will display an output of "3...2...1" onto an LCD screen mounted on the front of our robot to inform the subjects of the photo that a photo will be taken. In the case that Camerazzi collides with a wall or other object, it will rotate by increments of 60 degrees to the right until it senses a heat source, after which it will and continue moving towards the detected heat source. Lastly, recent photos will be transferred to a Google Drive folder every 5 minutes.

To guarantee a certain number of photos in order to fulfill our high level requirements, Camerazzi will have a limit of how long it will roam, t_d, without capturing a photo. Once the robot has travelled for x seconds, where x is the maximum dimension of the room divided by the speed of the robot, Camerazzi will interrupt its normal pattern of behavior, because this signifies that it has travelled at least as far as one length of the room without detecting a face. At this point, Camerazzi will move straight until it hits a wall, turn away from the wall, and capture a photo regardless of face detection.

$$t_d = \textit{time taken to cross room's length} = max(l, w) / v$$

Below we dive deeper into the specifics of each subsystem which will allow us to achieve this overall behavior.

B. Roomba Movement

Our robot will be moving via the Roomba as its base, communicating with the Raspberry Pi over Mini Din. In doing so, we must account for collisions with any object or wall and redirect the robot after contact. For collisions, the Roomba has built-in sensors we are utilizing. The following table displays the sensors we will primarily be interacting with and the range of their values that we will receive.

Sensor	Range	Index
bumps_wheeldrops	[0-15]	0
angle	[-32768-32767]	12
light_bumper	[0-127]	35
light_bumper_left	[0-4095]	36
light_bumper_front_left	[0-4095]	37
light_bumper_center_left	[0-4095]	38
light_bumper_center_right	[0-4095]	39
light_bumper_front_right	[0-4095]	40
light_bumper_right	[0-4095]	41

Using these values, our robot can detect a wall or object, such as the leg of a chair or table, before it bumps into it. To avoid any force that may shake our robot structure, we want the robot to stop at least 1 cm away from any obstacle, including walls and objects such as the legs of chairs and tables. In addition, the robot must be rerouted by turning and moving forward after detecting a wall or object and avoiding collision.

The `bumps_wheeldrops` sensor's 0th and 1st bits are binary indicators for the state of the left and right bumpers. These should remain 0 throughout our robot's operation to avoid actually hitting a wall or object and shaking the robot. The angle sensor gives us the angle in degrees that Roomba has turned since the angle was last requested as a signed 16-bit value. Counter-clockwise angles are positive and clockwise angles are negative. The value returned must be divided by 0.324056 to get degrees according to iRobot documentation due to a known flaw in the angle returned from the Create 2 sensor. The `light_bumper` value gives a binary indicator of each of the light bumpers with its different bits. The 0th bit is "Light Bump Left Signal", the 1st bit is "Light Bump Front Left Signal", the 2nd bit is "Light Bump Center Left Signal", the 3rd bit is "Light Bump Center Right Signal", the 4th bit is "Light Bump Front Right Signal", and the 5th bit is "Light Bump Right Signal". From this binary, we can efficiently retrieve the actual strength of each bump by only polling for those that have a hot bit in the `light_bumper` value. Each of these bump signals uses IR to retrieve the distance between it and the object it's detecting.

In accordance with the flow of Camerazzi, our robot will use the IR light bumper sensor values to stop the robot. Then, it will turn to try a new path. Since the thermal camera sensors have a 60° field of view, the robot will turn right 60° to get a new view without overlap of the old view in order to scan for high human-like thermal signals. In essence, the robot will attempt to avoid detecting the same humans and taking their photos over and over. The robot should attempt to move towards a thermal signal to take photos of humans. If it repeatedly senses a wall, it will continue to turn to the right 60° at a time until it no longer senses a wall or object, and/or senses a human through thermal

readings, and has a clear path to move forward. It is not turning 180° and simply moving away from the wall or obstacle because this may result in the robot bouncing back and forth along the same vertical path in the room if its path just so happens to be unobstructed.

A major portion of our robot is detecting humans, for both taking photos of and avoiding collisions between our robot and humans. This section will outline the avoidance of collisions. (Refer to Photo Optimization Subsystem for description of detecting humans for purposes of taking photos.)

The robot uses both IR and thermal camera sensors for human detection, with the IR sensors' output pin connecting to an analog pin on the Arduino (GPIO) and the thermal camera sensors communicating over I^2C with the Raspberry Pi. There are 3 sets of sensors facing the front of the robot and 3 sets of sensors facing the back of the robot (more details about number of sensors in Trade Design section). With the 13-inch diameter of the Roomba, having one set of sensors on the left-most side, one set in the middle, and one set on the right-most side will be able to detect any human that is standing in front of or behind the Roomba, even if the human is sideways. After detecting something is in front of the Roomba, the values we get from the IR sensors tell us the distance the human is away. The goal is to stop at least 3 feet away from any human (refer to Design Requirements section). Knowing that these voltages correspond with these output values from the IR sensors through the Arduino,

Voltage	Output from IR sensor through Arduino
0 V	0
5 V	1023

then 1 V corresponds to a value of 204.6 form the IR sensor through the Arduino. The Roomba should attempt to stop when it detects a human 3.5 feet away, to allow for 1 second of latency at 0.5 ft/sec, and still stop at least 3 feet away from the human.

3.5 feet = 106.68 cm

From the IR sensors' specifications, we know that the voltage is 0.4 V at 150 cm and 2.5 V at 20 cm.

$$\frac{2.5\ V - 0.4\ V}{150\ cm - 20\ cm} = \frac{21\ V}{130\ cm} = 0.01615385\ V/cm$$

$$0.01615385\ V/cm \times 106.68\ cm = 1.72329272\ V$$

$$1.72329272\ V \times 204.6 = 352.58569 \approx 353$$

So once we get an output value of 353 from any of the IR sensors, the thermal camera sensor will then assist in determining if it's actually a human or if it's just a wall. If the data from the thermal camera tells us there's heat signature, the robot will stop and take a photo based on the Photo Optimization subsystem description. The camera has a 62.2° field of view, so after it takes a photo, the goal is for it to discover a whole new view in order to detect and take photos of new subjects. Thus, after it takes a photo, it will rotate 62.2° to start with a new field of view. The robot will alternate turning left and right 62.2° in order to avoid going in circles if no bodies are detected.

After a human is detected and the camera feed is analyzed, adjust according to instructions from photo optimization algorithm (see Photo Optimization Subsystem for details).

C. Photo Optimization

Another main subsystem of Camerazzi is the photo optimization algorithm which will determine when and where photos are captured. Specifically, our photo optimization algorithm will use data from the camera, thermal camera sensor, and IR sensors to determine whether it is in the right position to take a photo at a given time. As a result of the data, the robot will move forward or back, and the camera will move up and down using our motorized stick, in order to satisfy our requirements for ideal photos. The thermal and IR sensors will be used to identify when we believe there is a human in view of the robot. As specified in the section above, we will detect humans and stop, then activating our face detection algorithm to begin searching for faces. Using live video data from the camera, our photo optimization algorithm will use OpenCV Haar Cascades to identify if and where there are faces in view.

If faces are detected, our photo optimization algorithm will check multiple parameters to determine how to adjust and capture the photo. First, we will check if all faces are above the minimum width requirement of 10% image width. As a second check, we will determine whether or not there is enough margin (5% image width) around all faces. If either of these requirements are not met, we will adjust the position of the Roomba to attempt to satisfy them. For example, the Roomba will move forward if a given face width is too small, and backwards if the face width is too large. Similarly, if there is too little margin around a face to the left or right, the Roomba will move backwards, and if there is too much margin, the Roomba will move forwards. Lastly, if there is too little margin above or below a face, the motorized stick will move up or down accordingly. Due to the fact that adjusting to one constraint will affect the value of the other variables, we have come up with a protocol that will help us determine how the robot will move, or if it should continue moving because the requirements cannot be satisfied. In the order of vertical margin, face width, and horizontal margin, Camerazzi will adjust its position or camera height slowly until the requirement is satisfied. After the given requirement is satisfied, we will check if earlier requirements were made invalid due to the following adjustments. In that case, the ideal photo is not possible by our requirements, so Camerazzi will continue roaming. This protocol for adjusting the robot for ideal image capture provides a straightforward approach to handling edge cases and determining whether photos should be taken or not.

D. LCD Screen and Motorized Stick

Other components in our system include the LCD display and the motorized stick.

The LCD display is the robot's way of communicating to the people in front of it that it is going to take a picture. It will be connected to the Raspberry Pi over the I^2C interface and will go through the following sequence: "3!", wait 1 second, "2!", wait 1 second, "1!", wait 1 second, "SMILE!" By providing this feedback to the human, the human knows how long it should stay still for the picture.

The motorized stick will be used to adjust the camera's position to meet the 5% margin requirement on the top and bottom of the image. It will be attached to the bottom of the enclosure containing the Raspberry Pi camera and push it up when the bottom margin does not meet the 5% requirement. Likewise, it will pull the enclosure down when the top margin does not meet the 5% requirement. The Raspberry Pi will control the motorized stick over Bluetooth.

E. Power

The iRobot Create 2 will be powered by its accompanying battery, which has a battery life of 3.5 hours.

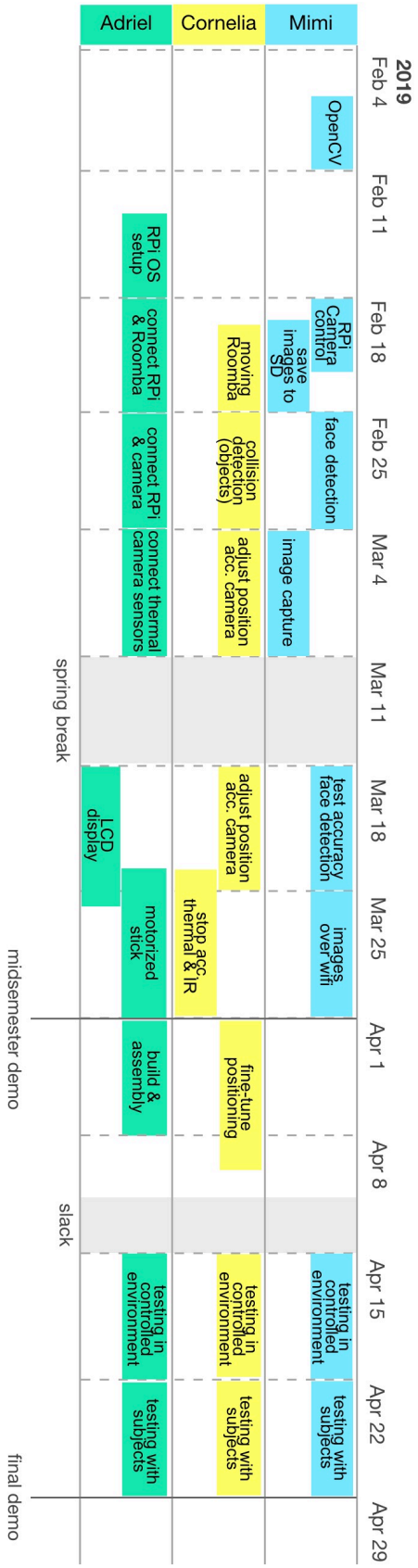
The Raspberry Pi will be supplying power to the sensors, the cameras, and the Arduinos. The Raspberry Pi itself will be powered by a 10000 mA/h battery. With the Raspberry Pi and camera module with Wi-Fi and Bluetooth on using 550 mA, the 2 thermal sensors using 100 mA each, the 3 Arduinos using 50 mA each, and the 18 IR proximity sensors using 33 mA each, we find total consumption to be 1494 mA. With an ideal version of the battery, we can provide power for a little bit over 6.5 hours. With a version that is 50% efficient, we will still be able to power our devices over 3 hours which is sufficient for our purposes.

Provide one or more overall system and subsystem figures. This should drop into more specific detail compared to the functional diagram described in section III. Specific chips, sensors, interconnects, etc. should be described in this section.

Concisely describe and, if appropriate, depict each major subsystem.

VI. Project Management

A. Schedule



B. Team Member Responsibilities

Mimi

As a software engineer, Mimi is responsible for the face detection portion of this project using OpenCV and NumPy on the Raspberry Pi. She will also do image analysis to determine the margins of the current frame and send that data to Cornelia to move the base of the robot for adjustments. After movement, Mimi will again check the margins and then ultimately capture the photo. Then, Mimi is responsible for transmitting the photo to a Google Drive folder over Wi-Fi.

Cornelia

As a software engineer, Cornelia is responsible for the Roomba movement portion of this project. Using PyCreate2, an iRobot Roomba opcode library, she is programming the Roomba's initial movement, processing thermal and IR sensor data for human collision detection in order to stop, and processing face detection/margin data from Mimi to adjust the Roomba's position.

Adriel

As a hardware engineer, Adriel is responsible for setting up the Raspberry Pi and Arduinos. More importantly, he is responsible for all the interconnects between the components of this project. He will also be working on building and assembling the physical structure of the robot, including the camera and LCD display enclosure, the attachments between the Roomba and the robot's top, and the extending limb holding the camera. This will also involve coding the stick to move up and down.

C. Budget

Part	**Source**	**#**	**$**	**Total $**	**Purchased?**
Raspberry Pi 3 Model B+	Amazon	1	42.99	42.99	Y
Raspberry Pi Camera Module V2	Amazon	1	24.68	24.68	Y
Arduino UNO	Amazon	2	11.86	26	N
Sharp GP2Y0A02YK0F IR proximity sensor	Robot Shop	17	13.02	221.34	N
Adafruit AMG8833 Thermal Camera Sensor	Amazon	2	43.99	87.98	Y
Micro SD Card	Amazon	1	11.39	11.39	Y
Tripod	Amazon	1	14.99	14.99	N
Wood	Woodcraft Supply	1	10	10	N
Motorized Stick	Amazon	1	21.99	21.99	N

Parts that were acquired without purchase:

Part	**Quantity**	**Contributor**
iRobot Create 2 w/ USB serial cable	1	Robo Club
Arduino UNO	1	18220
Sharp GP2Y0A02YK0F IR proximity sensor	1	Ideate Room
Power bank	1	Adriel

Tools we will be using to accomplish our project:

- Soldering iron
- Laser Cutter
- Sublime
- Visual Studio Code

D. Risk Management

A major design risk for our project from this point forward is the accuracy of face detection using OpenCV. Because the detection requires that the face is facing straight on with the camera, we may have problems with too few faces detected, resulting in too few images being taken. We have a plan to mitigate this using a high-level requirement of the minimum threshold of the number of photos being taken at an event based on the number of people in the room, length of duration of event, and length and width of the event space. We can also mitigate this risk by using additional OpenCV datasets, but this may reduce the speed of the video analysis and photo capture processes.

Another design risk is tables and chairs in the room that may be obstacles for our robot but above the height that the Roomba can detect it as such. In this case, we may need to place IR sensors up and down along the legs of the tripod to detect obstacles at all heights. With a durable robot structure above, however, we may be able to mitigate this issue by bumping into the table or chair and detecting the base using the Roomba's built-in light bumper sensors. Another design risk is the up and down moving mechanism for our camera enclosure to adjust for top and bottom margins in the photos being taken. The current implementation uses a motorized selfie stick that communicates with the Raspberry Pi over Bluetooth, however, because this piece is so mechanical and unique to our project in that we need it to be able to support our camera and Raspberry Pi enclosure, we may need to consult peers and professors in other departments to design something custom.

Schedule has been tight because we received our parts pretty late. Many things came up that delayed some of our progress such as the first iRobot Create 2 we borrowed from Robo Club was faulty and would not charge, the SD card that came with our Raspberry Pi was too small for OpenCV to be installed so we had to reorder it, and we discovered that just using thermal sensors would not give us the depth data that we needed so we had to reconsider our design and add IR sensors to our implementation. However, we are still on track with our Gantt Chart.

We have no problems with our budget and still have leeway if we need more sensors. In terms of personnel, we have just the right amount of software and hardware specialists to complete the tasks this project requires.

VII. Related Work

A similar project we found when doing research for our project is called the "Roomberry Surveillance Robot," which uses a Roomba, and Raspberry Pi Zero, and a camera, to send video through a web interface [1]. This project has many similarities to our project in terms of the hardware components it uses, and the functionalities it achieves such as a movable camera, saving photos, and serving up photos through Wi-Fi. However, this project lacks a lot of the more complex features we hope to include, such as optimizing photos and the Roomba position for human faces, and taking care of collision detection.

A similar product we found on the market is the "Double Robotics Double 2 Telepresence Robot," which is used for virtual interaction with remote individuals [2]. This robot is similar to our robot in shape and size, as well as the capability to take photos with a 5MP camera. However, this telepresence robot varies drastically in terms of cost and movement, as it is priced over $3000, and must be remote controlled for movement.

A product with a similar use case to our project is called the "Polycam Player," which was developed by Nikon company MRMC, and is designed to take the place of cameramen who specifically capture action shots in sports settings [3] [4]. The system uses face and limb detection to track players and get close up action shots and video of game play. This project shares many similarities with our system in terms of using feature detection to capture quality photos of humans.

Lastly, an article that we found, called "Autonomous Mobile Robotics Research for Daily-Life Environment," details the research of a university team that experimented with multiple types of robots in different environments [5]. This article was informative to us, due to its explanation of key functions and behaviors of indoor, autonomous, mobile robots.

References

[1] https://www.hackster.io/danimaciasperea/roomberry-surveillance-robot-Roomba-pi-zero-camera-c056f9
[2] https://www.bhphotovideo.com/c/product/1296165-REG/double_robotics_1012dr_universal_360_camera_mount.html
[3] https://www.technologyreview.com/the-download/610711/a-robotic-camera-system-films-sports-like-a-human-does/
[4] https://www.techradar.com/news/nikons-robotic-cameras-are-designed-to-give-sports-teams-the-competitive-edge
[5] https://www.sciencedirect.com/science/article/pii/S1474667017331099
[6] http://www.cs.cmu.edu/~illah/CLASSDOCS/p331-mumm.pdf
[7] http://www.gkstill.com/Support/crowd-density/CrowdDensity-1.html

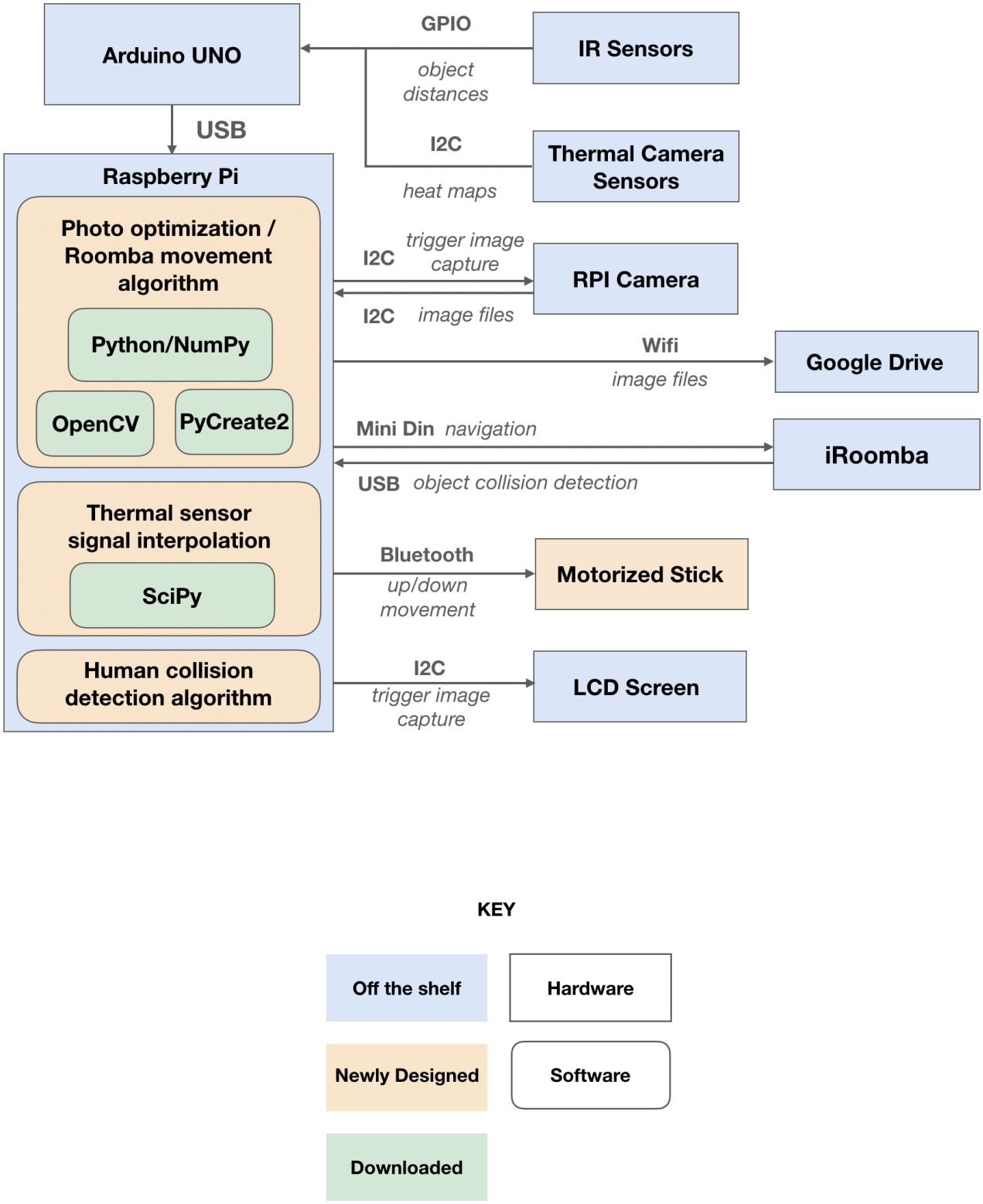

Arduino UNO
GPIO
IR Sensors
object distances
USB
I2C
Thermal Camera Sensors
heat maps
Raspberry Pi
Photo optimization / Roomba movement algorithm
Python/NumPy
OpenCV
PyCreate2
I2C
trigger image capture
RPI Camera
I2C
image files
Wifi
Google Drive
image files
Mini Din navigation
iRoomba
USB
object collision detection
Thermal sensor signal interpolation
SciPy
Bluetooth
up/down movement
Motorized Stick
Human collision detection algorithm
I2C
trigger image capture
LCD Screen
KEY
Off the shelf
Hardware
Newly Designed
Software
Downloaded

Fig. 1

User experience storyboard involving the host and guest users.
Place Roomba
Robot roams
Robot senses human
Robot adjusts position
Robot takes picture
Robot continues to roam room
Photos transferred to cloud
3.. 2.. 1..
SMILE!
PHOTO

Fig. 2