Smart Mirror

Author: Devon Barry, Christina Di, Judy Min: Electrical and Computer Engineering, Carnegie Mellon University

Abstract

As haptic technology is becoming more and more desired, research on interactive appliances has advanced tremendously. Smart mirrors are one such appliance that add ease and functionality to a user's daily routine. However, smart mirrors on the market right now such as Amazon's patented blended reality mirror and Lululemon's workout mirror are not easily accessible or affordable to users. Other smart mirrors on the market are expensive and require advanced setup. Ours is a smart mirror system capable of allowing users to try on tops by utilizing a computer vision torso and clothing detection algorithm that offers an affordable and accessible user experience.

Index Terms — Adaptive Content Generating and Preserving Network (ACGPN), Arducam, clothing detection, key point jsons, Jetson Xavier NX, OpenPose, torso detection, two-way mirror

I. Introduction

With the invention of amazon and e-commerce, more and more people are purchasing things online. Especially with the COVID-19 pandemic, our only option to buy clothes is online because it is unsafe to shop in stores. However, one of the major pain points of shopping online is the fact that we aren't able to try clothes on before purchasing. Many stores are final sale or just have really inconvenient return policies, so we end up spending money on clothes we may never wear. This is why we came up with a smart mirror. The smart mirror is designed to allow users to try on tops from home. Using a torso recognition algorithm, we will be mapping clothes from online stores or the users own closet onto the user's body with about an 56% precision and 4.5 second average latency to give the users a try on effect in as close to real time as possible.

There are many different smart mirror options available on the market. We drew a lot of inspiration from the workout mirrors that many workout companies have been releasing. In particular, we looked at MIRROR from Lululemon. Although the technology is extremely innovative and covers a great use case, the mirror is very expensive and doesn't solve the problem that we were looking to provide a solution for. We decided to also create a mirror, but we wanted our mirror to be cheaper and also help users shop from home.

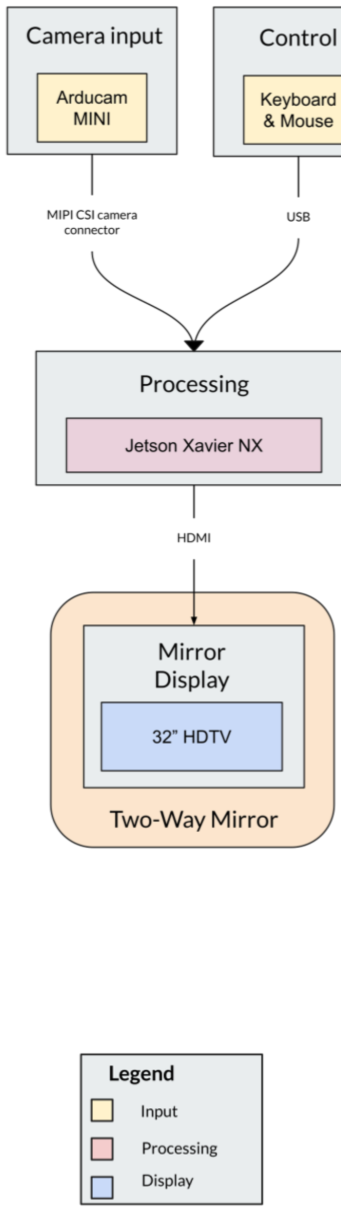
II. Design Requirements

As we hope to provide real time feedback to the user to give a true "try on" experience, it will be necessary to process images from the camera in real time. Thus, low processing time is a major objective for this project. Real time image processing is expensive, so we set a goal of 3.5 second overall latency between initial clothing projection as 4 seconds is the accepted time it takes before the average human being recognizes lag.

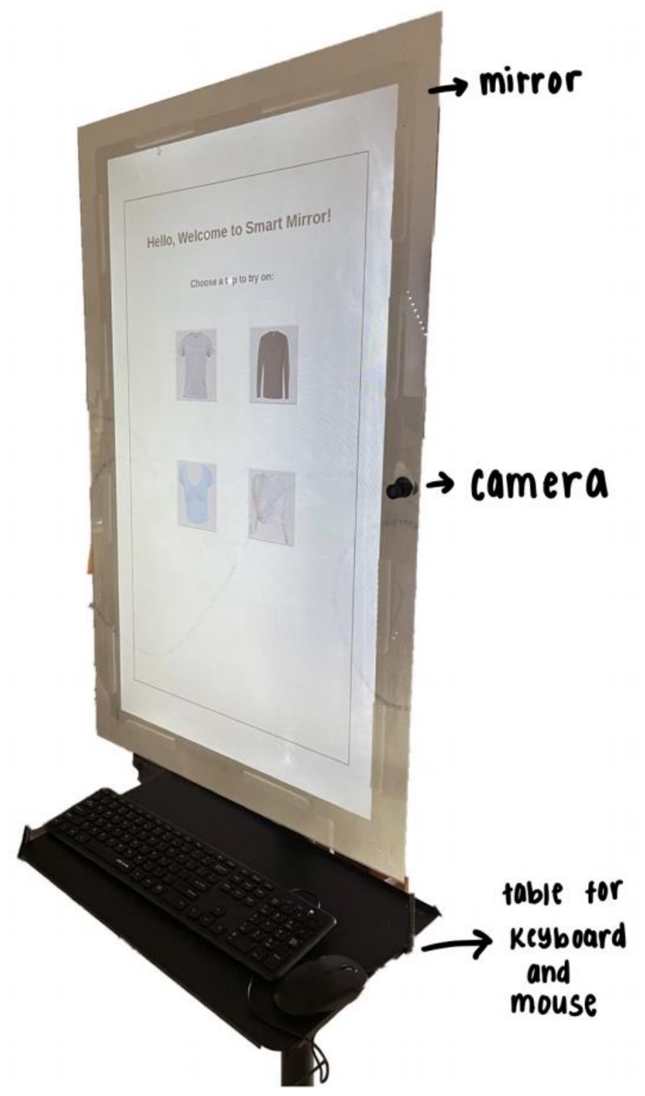
To cut scope, we are only performing torso recognition. Our minimum viable product includes the superimposition of a user provided image onto the image of the user. To measure the precision of our matching algorithm, we compare the key points generated from the algorithm to the recognized key points on models actually wearing the clothing. We count less than a 2-centimeter difference as a match. We expect 50% precision (or 50% fixed point matches) with superimposition and 80% precision with image warping. In regard to the off-the-shelf libraries we used for clothing and torso detection, we view these algorithms as all or nothing and expect 100% precision for both clothing and torso detection. This will be facilitated through providing an outline for the user to stand to limit user movement.

We require our image feed to be as close to life as possible. This is because we want what the camera sees to line up with what the user sees in the two-way mirror. We had originally planned to calibrate the images directly from the Arducam to make this as close to life as possible by adjusting the frame rate if needed. However, we found that calibrating the image directly from the input or the Arducam itself was cumbersome. Thus, we decided to calibrate the image to the mirror after running image processing and gathering the relative position of the shirt to the user. This gave us more room to use python image processing libraries on the backend and made it much easier to resize and adjust the image if necessary. It also allowed for saving the image as different image file types as well as merging the image with the background, all necessary parts to our final display.

III. Architecture and/or Principle of Operation



(a)



(b)

Fig. 1. System picture. (a) hardware block diagram (b) diagram of entire mirror system.

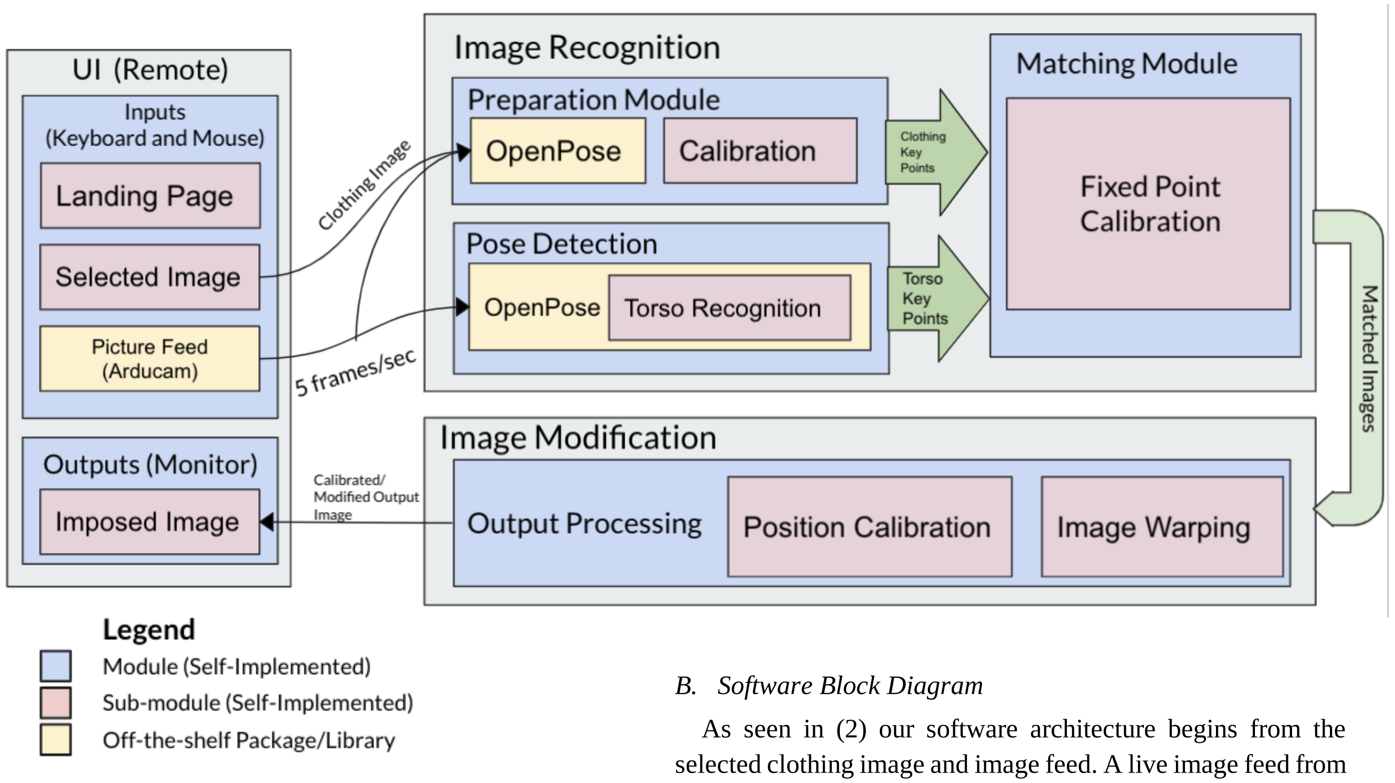
A. *Hardware Block Diagram*

As shown in (1)(a), our hardware architecture is relatively simple. The main aspects are input, processing, and display. We take the camera feed input of five frames per second via a camera and the user control input via a keyboard and mouse. The data from these inputs are sent to our processor, the Jetson Xavier NX, then the results are displayed on our monitor which sits behind a sheet of two-way acrylic mirror. All hardware parts will be mounted within a frame as can be seen in figure (1)(b).

The camera we are using is the Arducam MINI. The processor we are using is the Jetson Xavier NX, which is an embedded system-on-module from NVIDIA. The display monitor we are using is a 32" HDTV from Westinghouse.

We connect the Jetson Xavier NX processor directly to the Arducam Mini Camera via the MIPI CSI camera connector port. As our MVP, we will have a keyboard and mouse connected to the Jetson Xavier NX via USB. The display monitor will connect to the Jetson Xavier NX via HDMI.

Fig. 2. Software Block Diagram



B. *Software Block Diagram*

As seen in (2) our software architecture begins from the selected clothing image and image feed. A live image feed from the Arducam uploads five images per second to the Jetson Xavier NX. These images will be run through a pose detection module where OpenPose will perform torso recognition on the user. Then, the user selected clothing image goes through the clothing detection module, which also runs OpenPose on the clothing image. Both modules generate OpenPose key point jsons according to the 25 key point map.

The jsons from these two modules will then be run through a matching module where we will calculate the best positioning to place the clothing on the user's image. This is done through calculating the ratio of the center key point with respect to the image's original size and calibrated through resizing based on the display size. The matched image is saved to the Jetson Xavier NX and will go through an output processing module where we will calibrate the positioning of the display image to the mirror and optimally warp the clothing to the user before being sent back to the monitor where the output image is displayed.

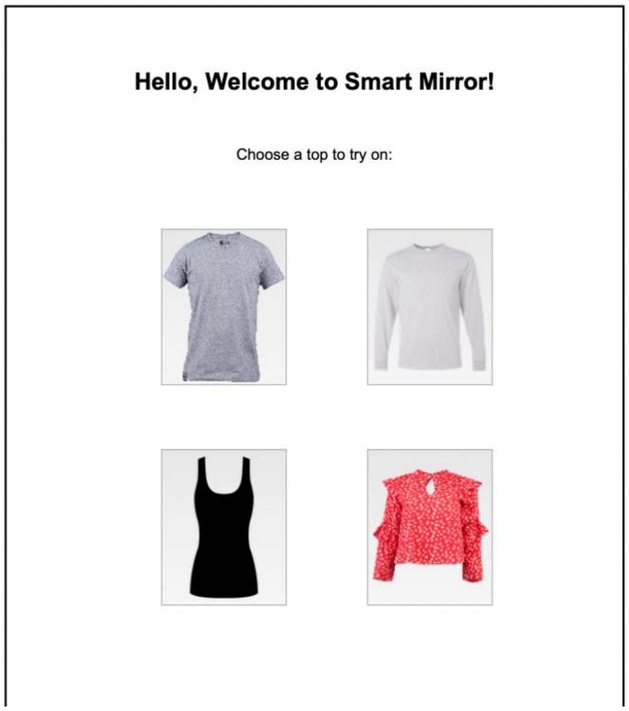


Fig. 3. User Interface Mockup

C. *User Interface*

We wanted to create a simple and user-friendly interface. Ideally, the images shown would have been scraped from a clothing website and background subtracted. As this was not in scope of our project, we opted for four preselected images that we know work with OpenPose and create usable key point jsons. The exact design of the user interface can be seen in Figure 3.

IV. DESIGN TRADE STUDIES

A. *Hardware Tradeoffs*

For hardware, we had a couple tradeoffs. For processing, we were initially contemplating using a Raspberry Pi for our project. The Raspberry Pi would have been sufficient for a simpler application, but since we are running a larger workload we needed a more powerful processor. This is why we turned to the Jetsons. We initially considered a Jetson Nano for its price point. However, in comparing the Xavier NX to the Jetson Nano, the Xavier NX is far faster – anywhere from two to seven times faster, depending on the application. The Jetson Xavier NX also comes with HDMI, USB and camera connectors, allowing us to connect to all of our other hardware parts in a simple manner. This makes the Xavier NX an obvious choice for our project.

For our camera input, we initially considered using two stereo Arducam cameras. These would have provided us with stereo depth input. We thought that this might help us with calibrating the image for the clothing mapping. However, upon further investigation we noted that the depth sensing provided by the Arducam stereo cameras did not offer depth resolution that would help us significantly with our project. We decided to go with the singular Arducam MINI camera for its size and its compatibility with the Jetson Xavier NX.

B. *Software Tradeoffs*

There really is only one feasible open-source choice for off-the-shelf torso and body detection, which is OpenPose framework developed in the CMU Perceptual Computing Lab. This library includes detection of the entire body including facial and finger detection. We were concerned that this unnecessary functionality for our use case may cause a runtime problem, so we were planning on subtracting that functionality for our purposes. Otherwise, the library is still quite heavy weight and is extremely slow on a laptop. After transferring OpenPose and running the image processing on the Jetson Xavier, the latency OpenPose did produce did not prove to be a priority to fix.

In terms of clothing detection, there are a number of classifier options all based on the DeepFashion2 [3] clothing dataset. We had originally selected DeepMark since it is the most accurate and lightweight package. Based on initial runtime testing, we switched completely to OpenPose. OpenPose is not meant for clothing detection, but still may work if uploaded images are well lit. This way, the key point generation from the clothing detection module will then match completely to the torso recognition module for easy matching and warping. However, going this route meant we had to compromise on our 100% requirement for precise clothing detection. This is a major tradeoff that would have introduced a new error flow to the design, but we solved the problem of having to implement this flow by using preselected images. It would have been helpful to use a true clothing detection algorithm such as DeepMark to test our matching algorithm’s effectiveness in terms of how well it fits to the body.

To cut scope and runtime, we also cut out the user uploaded image functionality. This way, we did not have to run a background subtraction algorithm on user images. This would have also introduced a new error flow that was out of scope.

C. *User Interface Tradeoffs*

As for the user interface, we had a couple of options that we were deciding between. At first, we wanted to make a touch screen smart mirror so the user could choose which tops to try on by directly touching the mirror. However, to make the screen touch screen, we would have needed an IR Frame. After adding the Xavier NX into our budget, we realized that we didn’t have enough money left to purchase an IR Frame. In addition to this, we realized that having the user walk up to the mirror to choose the top then walk back to try on the top would be a very awkward interaction, so we decided it would be better to use an external hardware as the controller for the mirror. We are going to use an external computer to create an interface which will send images of tops to the mirror directly.

In addition to the IR Frame, we were also considering purchasing a glass two-way mirror instead of an acrylic mirror. Although the acrylic mirror is a lot cheaper than the glass mirror, when we read the reviews, some people noted that there was a slight bend in the acrylic mirror which could give us issues with calibrating the screen and the mirror. However, we ultimately decided to stay with the acrylic mirror because with the Xavier NX, we would only have enough budget for one glass mirror. In the event that the glass mirror breaks or has any issues, we would have no way to purchase another. In addition to this, it won't affect our minimal viable product too much if the mirror is slightly bent. After doing a cost benefit analysis we ultimately bought the acrylic mirror.

V. System Description

A. Hardware

Processer

We chose NVIDIA's Jetson Xavier NX as our processor. As mentioned before, we chose this for its processing power, price range, and compact size. The Xavier NX is a machine learning platform and is designed for applications like ours which need greater AI processing power. The GPU in the Xavier NX is a NVIDIA Volta with 384 CUDA cores and 48 Tensor Cores. The CPU is a 6-core NVIDIA Carmel ARM v8 CPU with 6MB L2 cache and a 4MB L3. The Xavier NX comes equipped with 8GB of RAM.

Camera

We decided to use the Arducam MINI for its compatibility with the Jetson Xavier NX. The MINI uses the IMX477 camera module which is natively supported by Jetson. Because of this, we are able to use the camera with existing online open source code that uses the native camera commands such as nvgstcapture, nvarguscamer, or the nvargus API. The frame rate of the Arducam MINI is 1920x1080 @ 60fps and 4032x3040 @ 30fps. We used the be using the 30fps frame rate since this would be the most optimal choice considering the resolution of our monitor and our goal of minimizing end to end lag time for image processing.

Display

The monitor we are using is the 32" Westinghouse HDTV. This is a fairly standard LED TV with a 1920x1080 resolution. We chose the 32" size based on our requirements of torso detection. We would have liked to get a monitor that is truer to the size of a full length mirror, but it would have been far over budget. Therefore, the 32" is large enough to capture the user's torso. The depth of this monitor without the stand is 3.2" which makes it well suited for our purpose of acting as a mirror monitor.

B. Software

OpenPose

OpenPose is the first open-source real-time multi-person system to jointly detect human body, hand, facial, and foot key points on single images [1]. We utilize OpenPose only for torso recognition and do not need unnecessary functionality such as facial, finger, foot, and lower-body detection. As shown in (4) OpenPose generates 25 key point json which we will use to map torso to clothing with a focus on point 1, the center point.

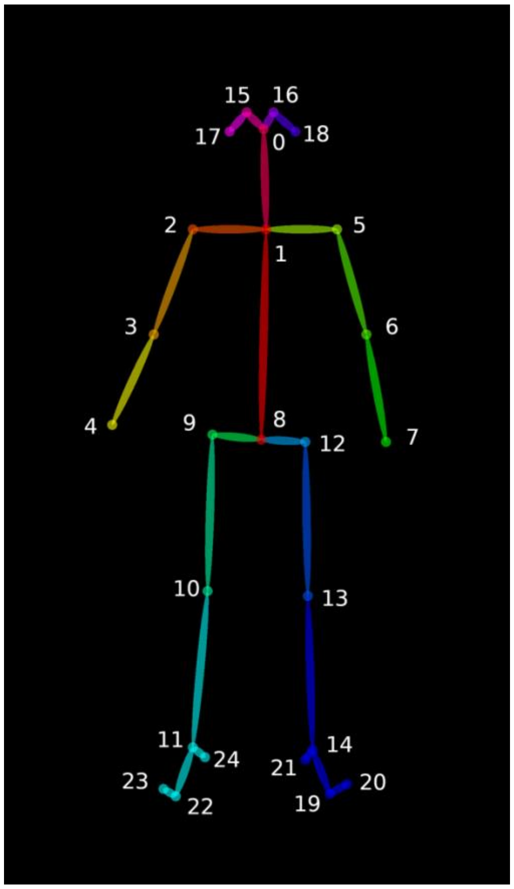


Fig. 4. Open Pose 25 Key Point Map

DeepMark [2]

DeepMark is a one-shot clothing detection algorithm that has state of the art accuracy of 0.723 mAP for bounding box detection task and 0.532 mAP for landmark detection task on the DeepFashion2 Challenge dataset. It runs optimally on low power devices but has very detailed and unnecessary classification of key points, which we would've had to match to key points from OpenPose torso detection. This made it a suboptimal choice for clothing detection algorithm. We had originally wanted to work with this software for clothing detection, but it actually made matching unintuitive. We still discuss its tradeoffs in the tradeoffs section.

Warping – ACGPN model

For our experimental clothes warping model, we experimented with using the novel visual try-on network, namely Adaptive Content Generating and Preserving Network or ACGPN for short [4]. We were curious about how we could use deep learning to warp the images of the clothes onto our users, making our mirror a more realistic try-on experience. ACGPN allowed us to incorporate this into our design. It is comprised of three modules: the semantic generation module, the clothes warping module, and the content fusion module (seen in figure below). The semantic layout generation module utilizes semantic segmentation of the reference image to progressively predict the desired semantic layout after try-on. We used the Self Correction for Human Parsing model trained on the LIP dataset [5] to generate a segmentation mask for this layer. The Arducam camera unfortunately did not provide great results for our segmentation masks, since the exposure was often times too low and the model was unable to detect a human form. In very bright lighting, however, it was more effective. Second, the clothes warping module warps the clothing images according to the generated layout. The last inpainting module integrates all the information for our final output.

After the user selects a shirt for the superposition and the camera capture the photo input, the image of the user goes into the clothes warping model pipeline. The image is cropped and resized, then fed through the segmentation model to create a segmentation map, and Open Pose to create a key-points file. The model uses this image, segmentation map, pose data, and clothing image to reconstruct a virtual try-on effect. The images are saved to a folder for the user to browse at any point.

We experimented with training the model ourselves using our own data but the results were not as satisfactory as the pertained model. For our implementation on the mirror, we used the pertained model to inference our images. While the warping model is too cumbersome for a full try-on effect, the user is able to keep a library of small preview images of what the clothes would look like warped to their body.

Matching Algorithm

This is a self-implemented module in which we will match the key points generated from the torso detection in OpenPose to the key points generated from our clothing detection algorithm. The robustness of this algorithm depends upon which clothing detection algorithm we end up using. If we use OpenPose to detect clothing, this will make the final positioning of the clothing much easier to isolate and match. If we end up using DeepMark, it will be much easier to perform background subtraction, but fixed point positioning will be more difficult. We will use a combination of the two and run the OpenPose algorithm over the background-subtracted clothing image to facilitate matching.

C. *User Interface*

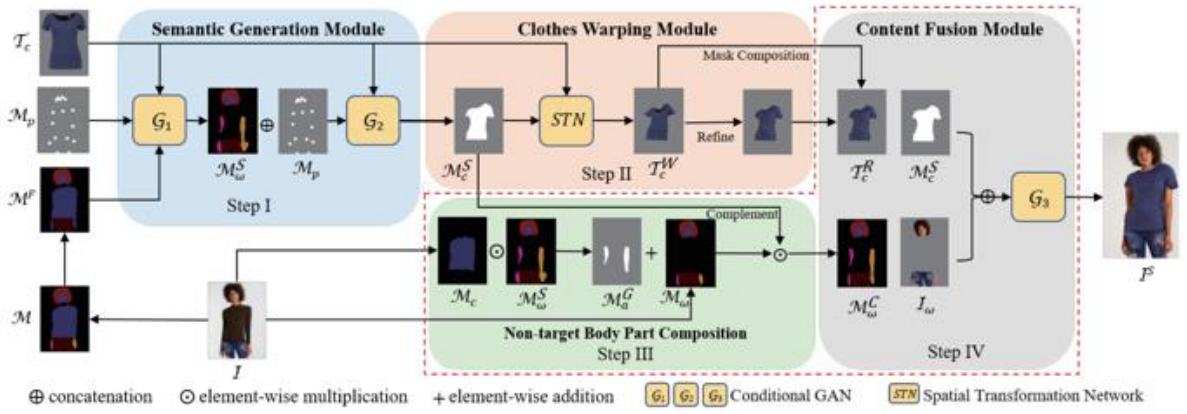
Keyboard and Mouse

The user will use a keyboard and mouse for remote control over the smart mirror.

Monitor

A 32-inch monitor will be used to display the processed clothing image. This will need to be matched to the user position, which will be tracked. Excessive user movement will be mitigated with a provided outline to suggest a user pose.

Fig. 5. ACGPN model flow



VI. Testing and Validation

All tests were performed on a data set of 2032 models images with 4 clothing images. Sample size = 2032*4 = 8,128 trials

A. *Latency*

Purpose and Requirement

We had originally set a goal for end-to-end image processing of 3.5 seconds latency.

Methodology

As OpenPose reports how long a process took, we use this in conjunction with a time decorator in our image processing python script to measure end-to-end latency for all trials. We take an average of these times to come up with a metric of 4.5 second average latency.

Results

After migrating OpenPose to the Jetson SDK, we found that our end-to-end average latency for image processing without any optimization on our part was 4.5 seconds, 1 second longer than our projected goal. We found this an acceptable base latency and determined it was not a priority to shave this one second off of the latency with optimization when our time could be better spent improving the matching algorithm.

B. *OpenPose Detection Precision*

Purpose and Requirement

Our system relies heavily on OpenPose performance. Complete OpenPose torso detection is absolutely necessary for even the most basic complete user experience. Thus, we require OpenPose to complete torso detection on our five frames per second without failure. OpenPose can handle if parts of the body are missing. However, our matching script assumes that the key points json generated by OpenPose is populated. We assume well-lit conditions where the torso is completely seen.

Methodology

We ran OpenPose on all 2032 of our model images and used a python script to examine the outputted key point jsons. If the key point jsons were populated as expected, then we count this as a success.

Results

The results of testing showed that OpenPose was able to recognize and generate a keypoint json for all 2032 model images used. We describe this outcome as 100% precision for OpenPose recognition.

C. *Clothing and Torso Matching Algorithm Precision*

Purpose and Requirement

Finally, we looked at the precision of our matching algorithm itself. We chose to use a semi-manual heuristic to provide more focus on the usability of our project. We want users to experience as much of a try-on effect as we can provide. We chose this mode of testing to measure how often we can provide this experience.

Methodology

Our matching algorithm was run on the OpenPose key point jsons generated from the 2032 images and the 4 clothing images from the UI. From here we manually select "passable" images. The criteria for passable images are as follows:

- *Center point alignment:* how well the center points of the model and shirt are aligned
- *Percentage of body shown:* we want a low percentage of the body showing outside of the shirt where it wouldn't be normally

We found that neck alignment is usually a good indication of our criteria. From these criteria, we were able to select passable and impassable images. We calculate precision from the number of passable images over the total number of images. Some example output images are shown in Figure 6.

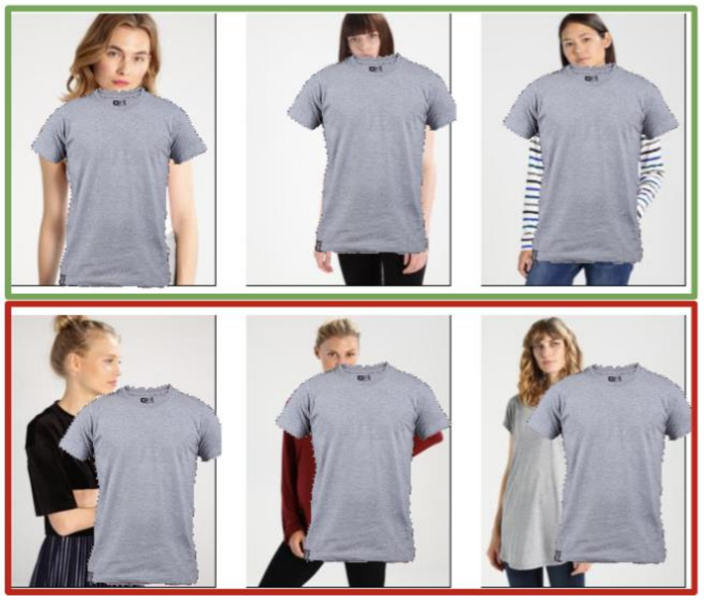
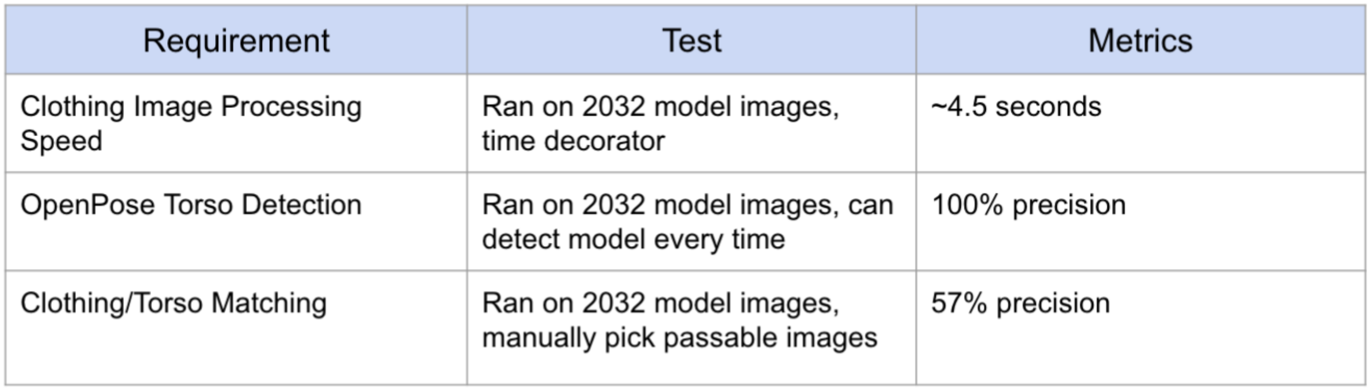


Fig. 6. Sample output images from our testing script. Passable images are boxed in green. Unpassable images are boxed in red.

Results

We averaged the precision of the 4 kinds of shirts we use on our mirror to come up with our overall metric of 57% precision. We can make a couple observations from these tests, notably that models that were angled away from the



Requirement	Test	Metrics
Clothing Image Processing Speed	Ran on 2032 model images, time decorator	~4.5 seconds
OpenPose Torso Detection	Ran on 2032 model images, can detect model every time	100% precision
Clothing/Torso Matching	Ran on 2032 model images, manually pick passable images	57% precision

Fig. 7. Summary of Testing Metrics

camera, had unconventional poses such as this one where the arms don't maintain a neutral position, or models that were sitting down consistently did not produce passable images. It's also important to note that we assume well-lit conditions where the model's torso is mostly seen. A graph of the 4 kinds of shirts we use on our mirror and how they performed is shown in Figure 8.

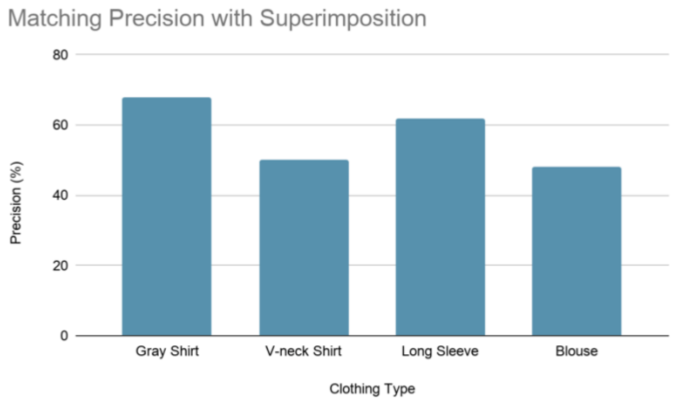


Fig. 8. Graph of matching algorithm precisions respective to shirt type.

VII. Project Management

A. Schedule

We divided our schedule into 4 phases. Phase one is the proposal phase. We allocated approximately three weeks for market research to develop the proposal for our smart mirror. The next phase is the design phase. We allocated three weeks to research the different technologies we want to use and also design the hardware and the system. The third phase is the implementation phase. During this phase, we spent 4 weeks implementing the torso recognition, clothes recognition, assembling the hardware, as well as creating the user interface. Then we moved onto the integration phase. We spent around 3 weeks integrating the different algorithms as well as integrating the software with the hardware. Lastly, the fifth phase was testing and demonstration. We spent the last three weeks testing to ensure that our algorithms are accurate as well as preparing the mirror for the demonstration.

B. Team Member Responsibilities

As In general, we will be working on many of the features together. However, Christina has taken a course on computer vision, so she will be focusing on the torso detection algorithm. Devon will be focusing on the clothes detection algorithm. Judy will be working on the user interface as well as assisting Christina and Devon on the various algorithms. We will all be working together on assembling the hardware as well as integration.

C. Budget

The budget is shown in Appendix B. We are borrowing the Jetson Xavier NX from Carnegie Mellon University, so the $399 will be taken out of our budget.

D. Risk Management

Our earliest risk posed was the cost of our design. We researched cost effective options as much as possible, but we still ended up a little bit over budget due to the expensive Jetson Xavier. Luckily after discussion with the ECE department, we can return the Jetson Xavier and keep this out of our cost, so now we have a lot of spare budget to use. We plan to keep it as a reserve for spare parts.

We're also concerned with the Jetson SDK compatibility with the software components, but there's pretty little documentation and we'll need to test it once it arrives. First priority on the software side is testing how the outputs of OpenPose and DeepMark correlate and we're in the process of doing that. We want to limit user movement as much as possible to facilitate successful torso and clothing detection so we'll have an outline for where the user stands.

Background subtraction stands as an issue currently for user uploaded images. We will need to isolate the actual clothing from the image so we can superimpose the image onto the user.

A true clothing detection algorithm will need to be used for this reason.

Monitor and mirror calibration is another risk. Because we want a live feed and live feedback, it's necessary that our camera sees what the user sees, so we'll facilitate this by adjusting frame rate of the Arducam if needed.

VIII. Ethical Issues

There are ethical issues that pertain to every type of product. For the smart mirror, a possible edge case for operation is that the user may be handicapped which could potentially result in a poor matching. If the user was missing a limb, during the warping process, the mirror may run into issues warping the sleeves to fit the user. However, the general matching of the shirt to the body would still be fairly accurate. OpenPose lists out the key points on the body that it detects. Any points that OpenPose can't detect are automatically set to 0, so OpenPose won't crash if it can't find something. In addition to this, the only point that we use for our matching is the center point. As long as the center point is detected, the user will still be able to try on the clothes.

Privacy has been a big area of concern for many of the smart mirrors on the market. The fact that there is a camera embedded into a mirror could lead to leaking of private images if the camera were hacked. However, the way our algorithm works, at one time we are taking and storing 5 images of the user to send to OpenPose for analysis and the matching. As soon as one analysis is done, the images are replaced with 5 new images. As soon as the user exits the mirror, those 5 images are immediately deleted making their private images safe from hacking. The only possible edge case is if someone hacked the mirror to send the images to a cloud or another database. Since the mirror doesn't need the internet to operate, this protects the mirror from many potential threats. If the WIFI were compromised, the mirror would be safe from any type of hacking. The only way to hack the mirror, the hacker would need physical access to the mirror.

IX. Related Work

There are many clothing detection algorithms trained on the DeepFashion2 dataset [3], but not many warping algorithms we could find other than the ACGPN model which we have utilized in this project. This is a relatively new area of research being explored for AR/VR applications as well.

X. Summary

A. Future work

If given more time, there are a couple of areas in which we could improve the smart mirror. First of all, ideally, we would like to allow the user to input clothes directly from online retailers instead of choosing from our pre-selected options. We needed some more time to find an image background subtraction algorithm. If a user inputted an image that they would like to use, we would have to get rid of any background or any people wearing the shirt so when we display the matching, only the shirt would show.

In addition to this, we would also want to expand the mirror to include bottoms in addition to tops. We had a limited budget, so we could only buy a screen big enough to showcase the torso of the person, but given more time and a larger budget, we would be able to do matching on the full body. This would allow the user to try on full outfits and even dresses and jumpsuits.

Lastly, we would do improvements on top of what we have now. We would be able to create a better and more user-friendly user interface. We would also improve the algorithm to fit in the 3.5 second latency goal we had originally set. We would also spend some more time training the model to improve it even further so the clothes would warp even better onto the user's body.

B. Lessons Learned

This was a great chance to dive headfirst into a project that we were all passionate about whilst using the skills we've garnered through our careers at CMU. While there were many technical challenges along the way, we were adaptable to change and were able to pull together a cohesive project that we are all proud of. Our team dynamic was effective and seamless, and it was really wonderful to work on such a supportive and consistent team.

References

[1] OpenPose Github, https://github.com/CMU-Perceptual-Computing-Lab/openpose#installation

[2] DeepMark, https://openaccess.thecvf.com/content_ICCVW_2019/papers/CVFAD/Sidnev_DeepMark_One-Shot_Clothing_Detection_ICCVW_2019_paper.pdf

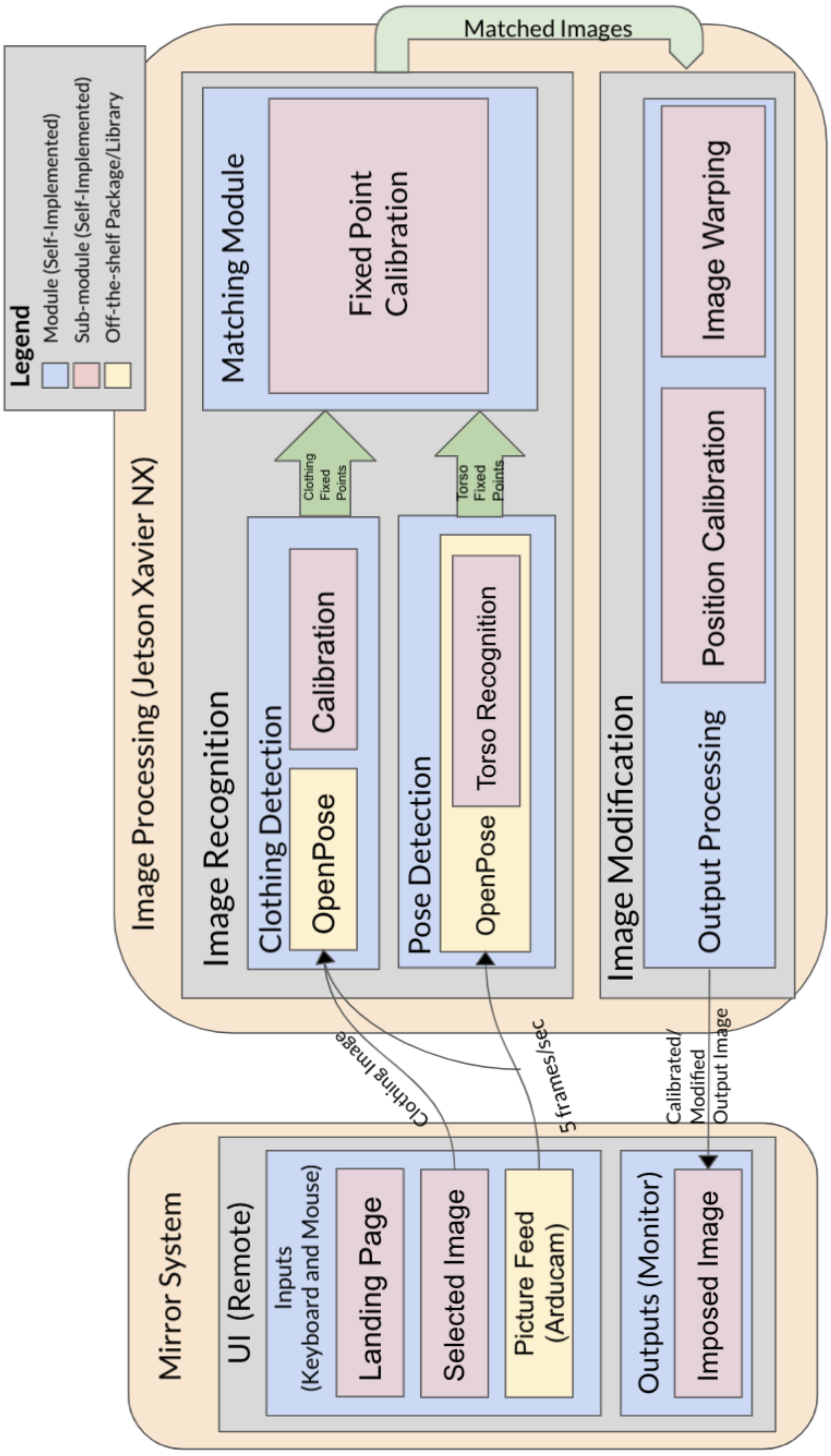
[3] DeepFashion2, https://github.com/switchablenorms/DeepFashion2

[4] Yang, H., Zhang, R., Guo, X., Liu, W., Zuo, W., & Luo, P. (2020). Towards photo-realistic virtual try-on by adaptively generating-preserving image content. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 7850-7859).

[5] Li, P., Xu, Y., Wei, Y., & Yang, Y. (2020). Self-correction for human parsing. IEEE Transactions on Pattern Analysis and Machine Intelligence.

[6] Rivers, Kelly. 112 Tkinter graphics template http://www.krivers.net/15112-s18/notes/notes-animations-part2.html

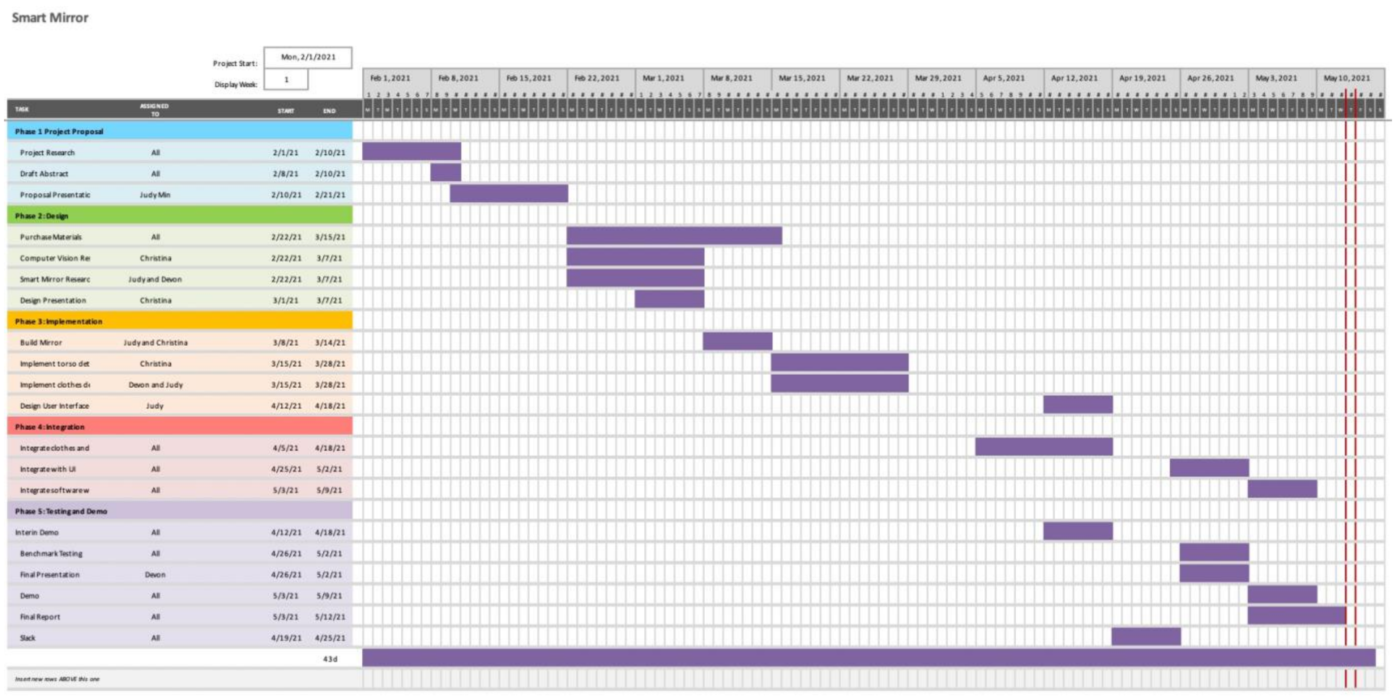
Appendix A


Legend
Module (Self-Implemented)
Sub-module (Self-Implemented)
Off-the-shelf Package/Library
Image Processing (Jetson Xavier NX)
Matched Images
Image Recognition
Matching Module
Fixed Point Calibration
Clothing Fixed Points
Torso Fixed Points
Clothing Detection
OpenPose
Calibration
Pose Detection
OpenPose
Torso Recognition
Image Modification
Image Warping
Position Calibration
Output Processing
Clothing Image
5 frames/sec
Calibrated/ Modified Output Image
Mirror System
UI (Remote)
Inputs (Keyboard and Mouse)
Landing Page
Selected Image
Picture Feed (Arducam)
Outputs (Monitor)
Imposed Image

Appendix B

Item	Specification	Price
NVIDIA Jetson Xavier NX Developer Kit	8 GB Memory	$399.00
32 inch Monitor	32 inch	$108
Arducam Mini	30 fps at 4K, 60 fps at 1080	$64.99
Two-way acrylic mirror	24 inch x 36 inch	$89.99
Total:		**~ $662**

Appendix C



Smart Mirror

Project Start: Mon, 2/1/2021

Display Week: 1

TASK	ASSIGNED TO	START	END
Phase 1 Project Proposal			
Project Research	All	2/1/21	2/10/21
Draft Abstract	All	2/8/21	2/10/21
Proposal Presentatic	Judy Min	2/10/21	2/21/21
Phase 2: Design			
Purchase Materials	All	2/22/21	3/15/21
Computer Vision Re	Christina	2/22/21	3/7/21
Smart Mirror Researc	Judy and Devon	2/22/21	3/7/21
Design Presentation	Christina	3/1/21	3/7/21
Phase 3: Implementation			
Build Mirror	Judy and Christina	3/8/21	3/14/21
Implement torso det	Christina	3/15/21	3/28/21
Implement clothes de	Devon and Judy	3/15/21	3/28/21
Design User Interface	Judy	4/12/21	4/18/21
Phase 4: Integration			
Integrate clothes and	All	4/5/21	4/18/21
Integrate with UI	All	4/25/21	5/2/21
Integrate software w	All	5/3/21	5/9/21
Phase 5: Testing and Demo			
Interin Demo	All	4/12/21	4/18/21
Benchmark Testing	All	4/26/21	5/2/21
Final Presentation	Devon	4/26/21	5/2/21
Demo	All	5/3/21	5/9/21
Final Report	All	5/3/21	5/12/21
Slack	All	4/19/21	4/25/21
			43d

Insert new rows ABOVE this one