Senior Design Project Report

Robotic Makeup Application System

Alejandro Varela

Jimmy Jason Ramsarran

Dillon Naidu

Jason Calle

Junmin Xu

Chun Kit Wong

The City College of New York

Department of Mechanical Engineering

Richard LaGrotta ME-47300

December 10th, 2019

Robotic Makeup Application System

Table of Contents

1) Executive Summary

The multibillion dollar cosmetic industry at an estimate of around $170 billion is an ever increasing and continually relevant field, not only today but also in the foreseeable future. For our senior project we were tasked with taking part in this industry by inducing a combination of innovative technologies to develop an apparatus that would automate cosmetic application. Specifically, the given problem is to design a system that will take a picture of an individual's facial structure and then have the system cosmetically modify the users face. For this project, the modifications will be performed on the user through utilization of a robotic apparatus.

To address the requirements of the given problem, we utilized a uArm Swift Pro robotic arm after multiple trade studies, in conjunction with Intel® RealSense camera. With a built-in Arduino Mega 2560 microcontroller in combination with a pressure sensor, the majority of the workload is based on the software and coding of these units for component integration, facial detection and capture, processing, and application. The final product is relatively mobile and can be readily mounted on a typical table of at least 2.5x2.5 ft. The user will be able to sit in a comfortable position within operational distance of the robotic arm's base. The working medium of either lipstick or ChapStick® is manually inserted into the arm's attachment holder, the user's photo is taken after the system is initialized, and the arm thoroughly applies the medium with no further human interaction required. The finish should be uniform and visually satisfying, not touching anywhere outside of the user's lips. The apparatus will be able to detect relatively slight movements of the user's face and compensate accordingly in a safe and efficient manner in between application loops.

2) Problem Description

Understanding the nature of the given problem is paramount as it should be with any engineering problem, though in this case it is readily apparent as a result of operating directly on a human's face. The assigned problem given does not dictate strict guidelines, procedures or requirements itself. We are required simply to design a robotic system that can detect and map an individual's face, after which cosmetic modifications are then applied. Additionally, we are required to ensure that our robotic system incorporates real time feedback.

In terms of driving requirements, safety is the prioritizing factor. A somewhat obvious requirement is that of user comfort and harm prevention; the user should not feel "threatened" by the movement of the arm and there must be absolutely no physical harm imprinted on the user. Keeping this in mind, the movement speed of the arm should be fast enough to complete the desired operation yet slow enough and free of erratic movements. If an individual reacts impulsively to any sudden or unexpected movements, the medium may be applied incorrectly or in undesired locations that could result in damage. Additionally it is important to both test and regulate the amount of force applied to the lips during operation which will require hypothetical assumptions and intuition, given the natural slight deformation of the lips and the amount of force or pressure that feels comfortable on that area.

With this basic set of requirements and the overall problem outlined, we analyzed the problem in much greater detail and formed a series of refined requirements which were used as guidelines for the development of our project. The table in *section 2.1* outlines these requirements along with a basic reasoning for each.

A) Requirements

Table 2.1.1. Requirements and Reasons

Priority	Requirement	Reason
1	No more than one user is allowed to be in the frame of the camera	Robotic arm will be unsure on which face to apply cosmostic material
2	The apparatus should be able to detect unsafe operational conditions through the use of sensors and react accordingly	User safety is of top priority
3	The software should be able to command the robotic arm to perform the desired operations	Without software functionality, the apparatus fails at its intended purpose
4	The system should incorporate real time feedback with respect to movements of the user's head/face.	To perform satisfying makeup application, the natural movements of the human body must be taken into account
5	The speed of the arm should not exceed 2 cm/sec to not intimidate the user	Estimate necessary for the material properties of the applicative medium
6	The user should have an easy-to-access user interface for makeup design selections	An interface would be ideal for visualization and style selection
7	A mirror or form of progress indication	For ease of use

8	The user faces the camera	The facial recognition works best when the person is facing the camera
9	User does not wear glasses, or should remove their glasses prior to use	Glasses may pose an issue when the robot moves near the eyes
10	Room is well lit	The camera sometimes has trouble detecting darker skin tones in lower light

The requirements listed in *Table 2.1.1* are listed in order of importance, with safety given the utmost priority since a well-designed and highly efficient product can still be rendered absolutely useless if it causes any form of harm to a client. For the specifics of this problem however, the area of focus is strictly on the lips which negates the need to worry about the more dangerous possibility of causing eye damage. Should our design expand into a full cosmetic routine in future stages, these parameters will need to be considered carefully. The criticality of this requirement governs every design choice and decision that follows. With respect to performance, the operation should be relatively fast in comparison to human application timing and also allow for the user to move slightly during the application, without having such movements hinder the outcome.

B) Interfaces

For our system to be user friendly, it is desirable to have some sort of a user interface through a computer program. The user should be able to have their makeup or chapstick applied with minimum human instruction or help. The software will then perform its purpose with a configuration molded specifically the user's 3D model of their face, which is accomplished by the use of the base-mounted Intel® RealSense camera. The camera will work

with the uArm Swift Pro through a laptop that is capable of running python code, which is the primary language used in the programming process of this project.

3) Trade Studies Summary

In the selection of components and parts used to design our project, we researched a variety of sensors, arms and controller boards for comparison. Using multiple categories of interest for comparison, we adopted a scale of 1 to 3 for scoring with 1 being the lowest and 3 being the highest. These results of these trade studies are shown in the following tables.

Table 3.1.1 Robotic arm trade study

Weight	3	3	3	2	2	2	2	2
Robotic Arm	**Affordability**	**On-board software**	**User-friendly**	**Degrees of Freedom**	**Repeatability**	**Expandability**	**Appearance**	**Compactness**
uArm Swift Pro	2	3	3	2	3	3	3	3
Universal Robots UR3	1	3	3	3	3	2	3	1
Braccio	3	1	2	3	1	1	2	2
Weighted score:	**uArm Swift Pro**	Universal Robots UR3	Braccio					
	52	45	36					

The uArm swift pro was chosen due to its overall outperformance or tie with the other arms, with the only negative being the price and one less degree of freedom. In all other areas the uArm receives excellent ratings and it was determined that this arm is ideal for the requirements and our project needs. Additionally, the uArm is popular amongst robotics enthusiasts on the internet, meaning there is a large pool of open source code and expertise. The robot is also designed to be expandable with a wide variety of sensors and robotic parts.

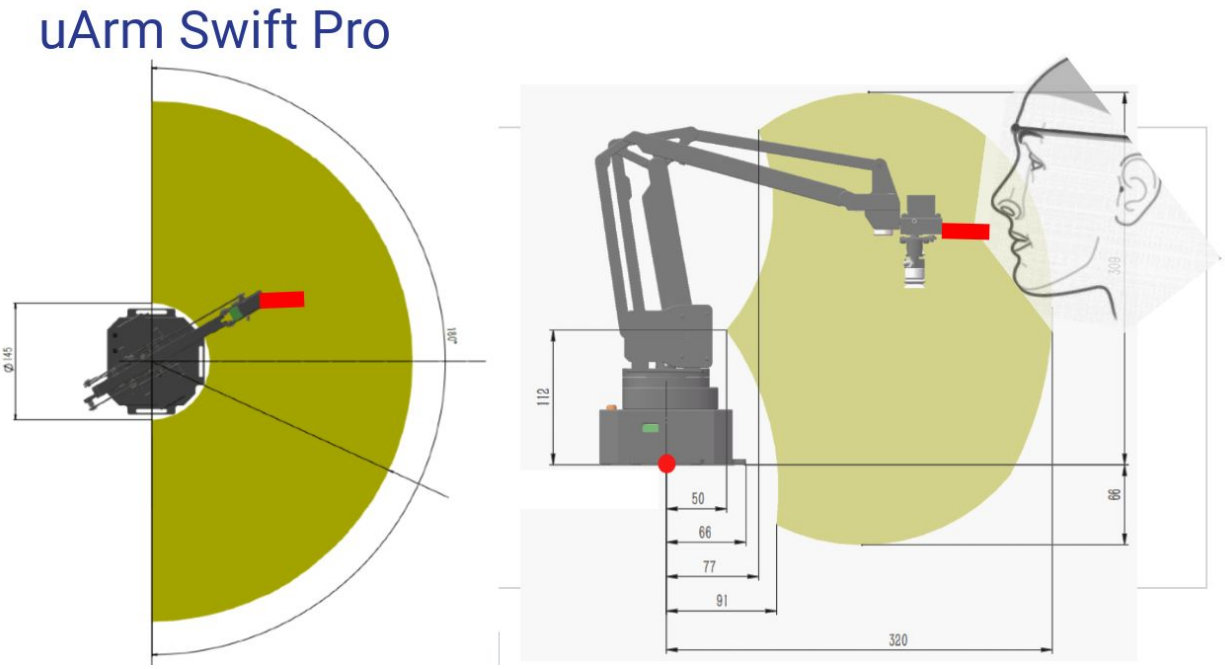


Table 3.1.2 Microcontroller trade study

Type of Microcontrolller	**I/O performance**	**CPU Intensive Performance**	**Web Support and Example Projects**	**Compatibility (with robotic arm and programming language)**
Weight	2	2	1	1
Raspberry Pi	1	2	3	1
Arduino	3	2	3	3

Weighted score	Raspberry Pi	**Arduino**		
	7	**11**		

When comparing the two major types of microcontrollers to use for our project, we decided to use an Arduino for a variety of reasons. The input/output performance is significantly more versatile in comparison to a Raspberry Pi, as is the case with compatibility with our chosen robotic arm. With respect to CPU performance and the availability of code to be found, they are relatively equal.

Table 3.1.3 Camera type trade study

Weight	3	1	2	3
Type of Sensor	**Programming ease**	**Depth tracking**	**Image sensing**	**Affordability**
OpenMV Cam H7	3	1	3	2
Pixy 2	2	1	2	2
Intel RealSense Camera	3	3	3	2

Weighted score	**Intel RealSense Camera**	Pixy camera	OpenMV Cam H7	
	24	17	22	

For selecting the type of camera to fit our 3D face processing and tracking needs, the initial choices were between the OpenMV Cam H7 and the Pixy 2. The research proved that it should be slightly easier to program the OpenMV, in addition to it having better image sensing capabilities. With all other areas at a tie, it was determined that an OpenMV Cam H7 was ideal for our needs. However, after consultation with uArm on their product and recommendations along with observing our budget allowances, we felt that Intel's® RealSense camera was more fit for this project and our design was then modified to use it.

Table 3.1.4 Programming language trade study

Trade Study: Programming language	**Experience with Language**	**Compatibility with Microcontroller**	**Accessibility (in computer labs)**	**Installation Complexity**	**Ease of use**
Weight	1	1	1	1	1
MATLAB	3	2	3	3	3
Python	1	3	2	3	2
C++	2	3	1	1	2
Weighted score:	**MATLAB**	Python	C++		

	14	11	9		

As integral as the components and the assembly of components, is the programming behind the entirety of the operation. With this in mind, a trade study on the ideal form of programming language was necessary. The clear winner in this trade study was MATLAB, due in no small part to the experience, accessibility and ease of use for this group. Although Python and C++ are more ideal when it comes to using Arduino microcontrollers, we assumed that our lack of experience with the languages may result in compiled complications and as a result, MATLAB became the apparent choice for our programming needs. As we began development of the integration and the coding however, we realized that despite the outcome of our trade studies Python was more suited towards our needs. This was because both uArm and Intel released software development kits for their products, and Python was a programming language that both products supported. Also, both companies provided example coding files which allowed us to gain a better understanding of both Python and how to operate the devices.

4) Summary of Solutions

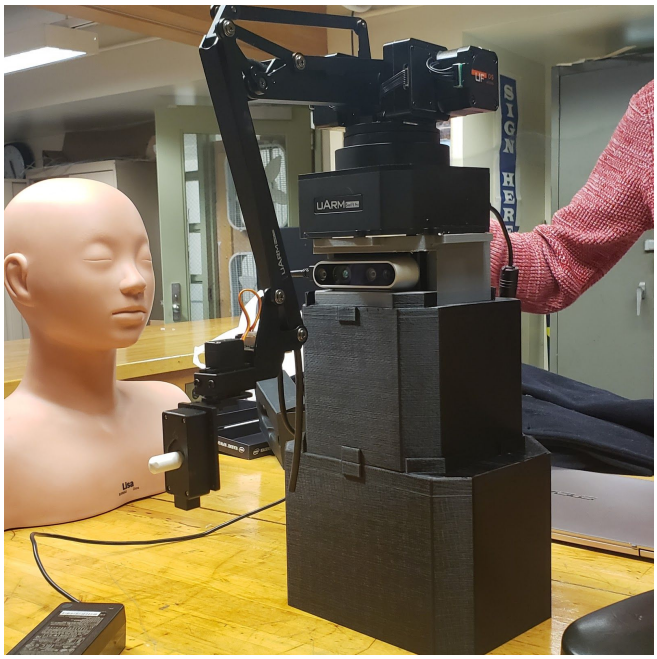


FIgure 4.1: The completed apparatus including the uArm Swift Pro, Intel RealSense Camera, and the 3D printed base.

For our approach to the given problem the bulk of our components were purchased and assembled together with relative ease, with the programming and integration of components proving to be the greater challenge. The governing thought process involved a solution in which we minimize the production cost of our apparatus/product, but only to an extent where the requirements are met satisfactorily and where the operation of our final result is safe and efficient. The design utilizes a robotic arm made by the robotics company uArm: the uArm swift Pro model, and an Intel® RealSense depth camera. A force sensor, controller and universal holder as an end effector— all manufactured by uArm— were also chosen for our design.

Operationally, the apparatus first takes a picture of the users face when they are within range (operation is manually initialized through running of the code) of the arm, which is at about 1 ft or 320mm away from the camera. The user must be seated in a chair with the midsection of their head ideally in line with the camera and with the apparatus firmly on a table. Once the desired makeup type is attached to the universal arm holder, the physical and autonomous operation begins when the code is run. An image obtained from the Intel®

RealSense camera is used for the application of the facial landmarks. Facial landmarks are used to pinpoint 68 distinct locations on the face, using libraries provided in Python software. By mapping the landmarks in a loop, a real time view of the 68 points can be obtained as the picture is refreshed quickly enough to simulate a video. Using the information obtained from the Intel® RealSense camera, depth information is provided alongside the coordinates given from the landmark tracking. This gives x, y, and z coordinates which the robotic arm can be set to move towards. This operates in conjunction with the depth sensing capabilities of the camera, which gives the system a distance reading between the robot's base and the user. If the user steps out of range of the camera, the system will pause and resume only when the user is once again within range.

For the arm and camera to be of sufficient height and so that the user is not required to bend at an angle for idealized detection, it was determined that a multi-part base should be constructed for the arm to rest on and so that the camera is held at a fixed position. The final design was a 3 piece assembly constructed in SolidWorks that is held together by support tabs as shown in the base drawings in *section 3.3*. For added support to prevent slippage since the pieces are constructed out of plastic, thin rubber sheets were placed underneath each base section. The 3 piece design was chosen to facilitate different user heights or abnormal seating/table conditions.

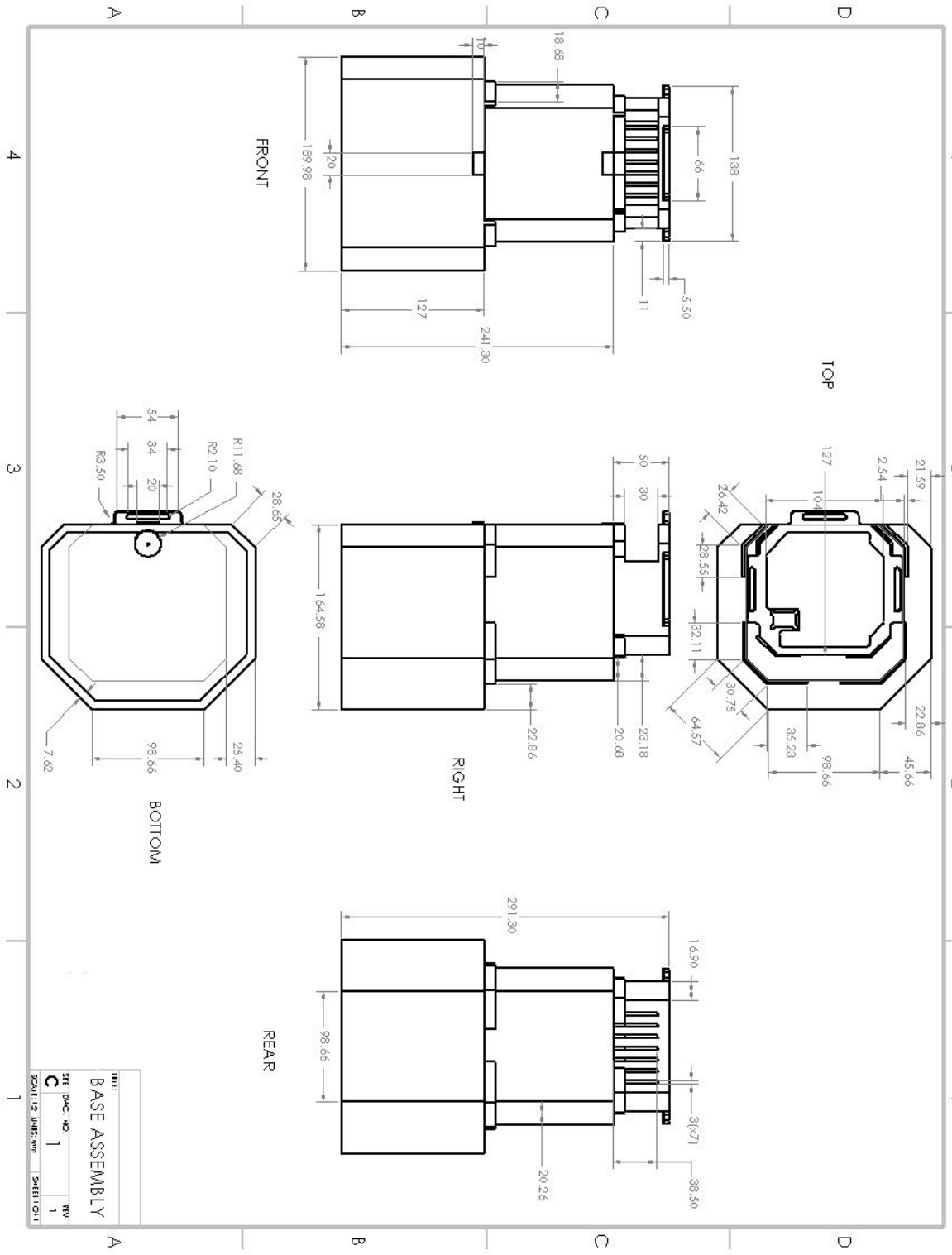
A) Cost Analysis of product design

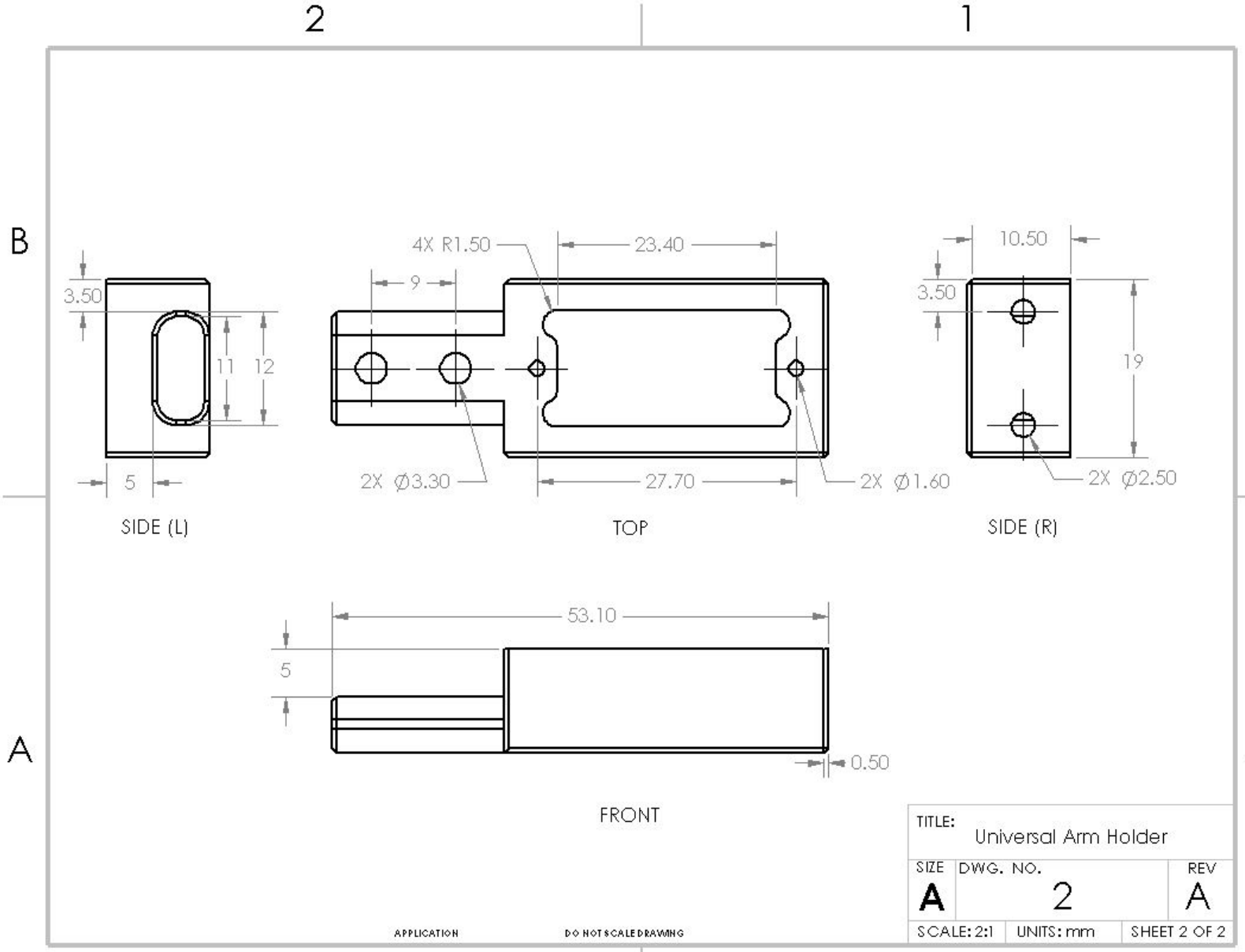
Table 3.2.1 List of components and costs

Part	Cost $	Link
uArm Swift Pro	749.00	https://store.ufactory.cc/products/uarm?variant=22044585328758
uArm free of cost	-749.00	
Intel RealSense Depth Camera D435i	199.99	https://store.intelrealsense.com/buy-intel-realsense-depth-camera-d435i.html
uArm Universal Holder	25.00	https://store.ufactory.cc/products/uarm-universal-holder
uArm Ultrasonic Ranger	19.90	https://store.ufactory.cc/products/uarm-ultrasonic-ranger
uArm Controller	99.00	https://store.ufactory.cc/products/uarm-controller
3D Printed Base	68 .00	(Printed at city college)
Rubber Sheet	9.66	
Total	$411.89	

The budget assigned to us was not fixed, but it was desired that we keep costs at or under $1,000 so that we left room for extra purchases or unforeseen events. The component that took a significant portion— around 75% of the allocated budget— was the robotic arm. After confirming our choice as the result of the trade studies, we debated this purchase due to the lack of reserve funding after purchasing the remaining components. Thankfully, we were able to win the exact arm in a giveaway event on social media which freed up our budget, allowing us to choose a better and more expensive camera which improved the performance of our design. With the new overall expenses of $411.89 before shipping and taxes, we have an estimated 59% of the proposed budget limit left, leaving us at a major advantage compared to our initial situation.

B) Manufacturing / descriptive drawings




2
1
B
A
3.50
11
12
5
SIDE (L)
4X R1.50
23.40
9
2X Ø3.30
27.70
2X Ø1.60
TOP
10.50
3.50
19
2X Ø2.50
SIDE (R)
53.10
5
0.50
FRONT
TITLE:
Universal Arm Holder
SIZE
A
DWG. NO.
2
REV
A
SCALE: 2:1
UNITS: mm
SHEET 2 OF 2
APPLICATION
DO NOT SCALE DRAWING

C) Design Solution

The Python programming was critical to the performance of the project because it provided a connection between the Intel RealSense Camera and the uArm Swift Pro. To use python on the laptop, Anaconda was installed as the choice of python distribution. This provided pre-installed libraries that were used for both uArm's and Intel's example programs in their software development kits. However, their software development kits were installed manually because they were not included in Anaconda. Also, two extra libraries were used for the facial recognition software.

Intel provided multiple example codes which allowed us to understand how to gather useful data. First, the camera's information was displayed using the image sensor and dual infrared cameras. The left image is the video output, and the right image is a depth-based image formed from the infrared cameras. The images were initially not aligned, but they were programmed to be aligned with help from the example codes. This was necessary because points on the video camera were needed, and if the images were aligned, a depth value for that point could be obtained.

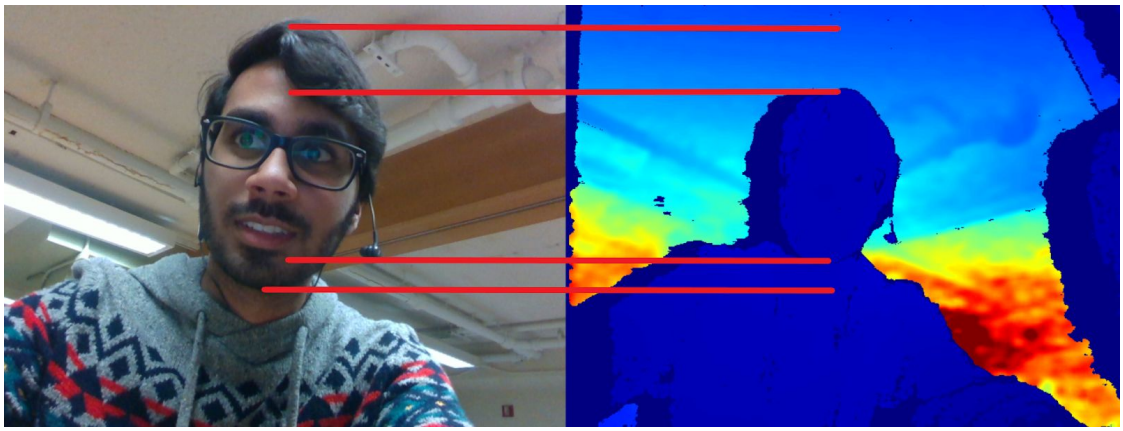
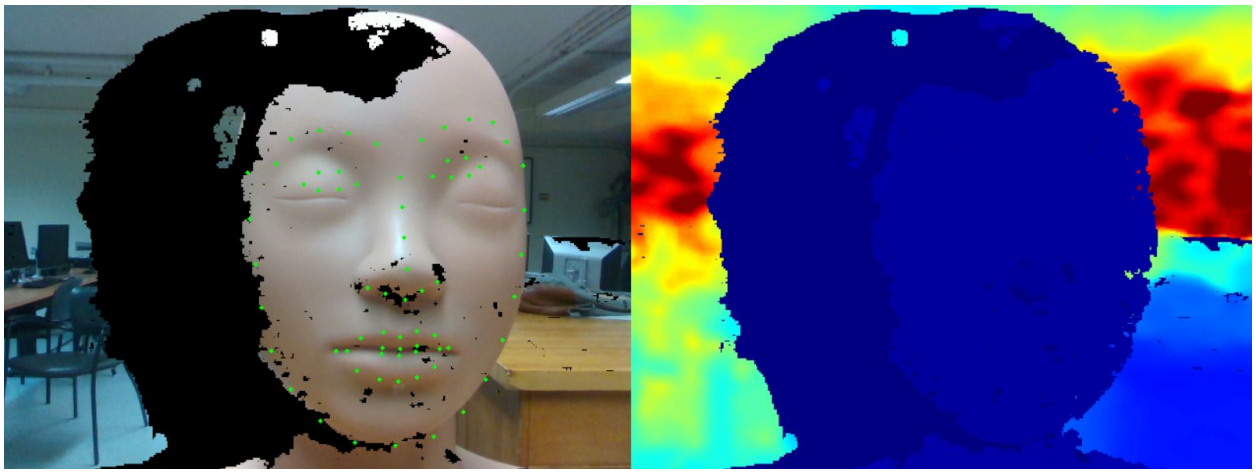


Figure 4.2: The unaligned Intel RealSense Camera output.

The facial recognition was provided using open source software found online. The code uses a machine learning library named dlib and a file containing the pre-trained machine learning data. Using this code, a set of 68 points can be applied to specific locations of any detected face. The points are applied to an image, but by repeatedly applying the points to

current camera data, real-time data can be observed. This gives the location of the face, and by combining it with the Intel RealSense Camera's output, a 3-dimensional coordinate of the person's face can be obtained. The landmarks are placed onto each image using a coordinate based on the pixels, then the third coordinate is obtained using the depth camera's depth at that coordinate. This gave a coordinate that must be converted to a coordinate system that the uArm Swift pro can use when programming its movement.



The goal of the conversion of the coordinate system was to map each camera coordinate to each coordinate in uArm's coordinate system. We found that uArm's coordinate system uses an origin at the base of the arm, and uses units of millimeters to move in x,y, and z distances from the arm. We used this information to test the range of motion of the uArm Swift Pro. We observed the limits of the arm's movement inside the range of the camera, then noted the coordinates used to reach those positions. We also found the number of pixels horizontally and vertically for each frame given by the camera.

Axis	uArm movement	Pixel Count	millimeter range of motion
X	Forward and backward	N/A (there are no pixels associated with the depth)	100 < x < 300
Y	left to right	640 (x coordinate of camera output)	-150 < y < 150
Z	up and down	480 (y coordinate of camera	-150 < z < 150

		output)	

For the horizontal motion, the total millimeter distance between the positions were divided by the number of horizontal pixels in the frame, then multiplied by the given camera coordinate. However, since the middle of the camera contained the 320th horizontal pixel while the robotic arm was at y = 0, the conversion had to take this into account by offsetting to be centered. Also, since the video camera of the Intel RealSense camera was not in the center of the sensors, a further offset was needed to get a proper alignment between the two coordinates. A similar process was used for the y-coordinate of the camera, which was the z-coordinate for the robotic arm. The x-coordinate of the arm was for depth movement, and it had a simpler conversion because the depth information given by the camera was in meters, while the arm moved in a coordinate system using millimeters. the value in meters was converted to millimeters then offset slightly backwards, to give room for the length of the chapstick to apply onto the lips without applying too much force on the lips. Once the method of converting all three coordinates were obtained, the 68 landmark points were converted into the coordinate system of the robot arm.

The uArm swift pro could be programmed to move to specific points without an issue, but the three degrees of freedom of the robotic arm became problematic. The robot arm pointed radially from the base, which prevented it from accurately applying chapstick on a lip. This was because as the arm moved away from the center, it would apply chapstick at an angle that caused it to be too far from the face. A servo was added to the end of the arm to provide the fourth degree of freedom. It began facing the arm at all times by programming the wrist of the arm to turn at an angle opposite of the arm's current angle. This greatly improved the arm's accuracy after initially having issues.

To program the arm to move to the points of the lips, the facial landmarks relating to the lips were used. The converted x, y, and z coordinates were used in a loop to get the arm to move along all points on the lips. However, just converting once was not enough, since the apparatus was required to track the face in real time. To accomplish this, the camera's

converted coordinates were stored in a file which was repeatedly checked during the movement of the arm. After the arm completes one loop around the lips, the arm moves away from the face to allow the camera to view the face again, and if the camera still sees a face, the arm will move to the new face, using updated coordinates. If there's no face in the camera's view, the robotic arm will not move to a face. This is because the program checks if the coordinate file was updated within the last two seconds, and only moves if this check succeeds.

5) Predicted Product Performance

Against the critical requirements, the main areas to consider are the force applied to the user's lips and the verification of the operational procedure. Assuming the arm performs as desired, the application force will be less than 1 newton at all times during skin contact. With user sitting upright in a chair at the edge of a desk or table and at a distance of around 0.3 m, the arm should first take an initial photo and then map the necessary points to perform the lip operation. After performing one complete loop covering both upper and lower lip the arm moves away from the user and takes another photo for a repeated application, to which the user is welcome to stay if he/she finds another application to be necessary. This entire process is initialized by the person who runs the code on the connected laptop, not the individual getting operated on.

With respect to the application of the cosmetic medium chosen, the final result should be an evenly distributed layer or layers of the medium that does not extend anywhere outside of the user's lips. If our final result is sloppy in this or any related sense, it will count as a mission failure. Any unexpected arm movements or impacts to the face will be treated similarly which is why thorough testing is of utmost importance.

6) Testing

Testing our design to see if our requirements are met satisfactorily is more of an "visual output analysis" type focus as opposed to a numerical force or strain study. That said, the main forces under consideration are the forces that the makeup applicator/medium (lipstick or chapstick) subject on the arm of the robot, and the forces that are applied to the face of the user. The force or weight of the applicator with respect to arm length, angular acceleration and angle of contact does not matter as long as the applicator weighs 500 grams or less [1]. With this limitation set by the manufacturer, adhering to this restriction negates the need to do any type of force calculation or testing with respect to what is placed in the robotic arm universal holder.

The primary force that needs to be considered is the applied pressure to the human face during makeup application. While a very slight varying force is necessary and unavoidable, it should be limited to a level where human discomfort can never occur. To conduct such a delicate form of experimentation, we allowed our arm to run repeated trials of lip tracing on a realistically sized mannequin head. During the trials, we observed for any excessive head movement which would indicate the occurrence of too much applicator force on the face. One instance of excessive force would count as an automatic trail failure, since the user should never be subjected to discomfort. In addition to head movement we observed the accuracy of the actual application using white chalk for a higher visibility. If our robotic arm applies even a faintly visible mark on a surface that should not be touched, that trial will also be counted as an automatic failure. If we observe the presence of at least one trial failure after performing around 10 repeated trials, the apparatus cannot be fully verified as safe for human operation and it will require tuning of some sort.

To test the apparatus for handling of head movement during operation, we moved the mannequin head to five different locations within the range of the arm. We placed a red dot on the mannequin head's face to serve as the intended point to move the robotic arm. We used our conversion to get the x, y, and z coordinates, then moved the arm to the red dot to test how close the arm got to the point. Then, we used uArm's own software to manually move the

robotic arm, which would give x, y, and z coordinates at any point that the arm was moved to. We manually moved the arm to the red dot, and we observed the expected coordinates of the arm this way. We compared the actual values obtained from the camera with the values obtained from the robot arm, and obtained the average offset that the arm moved during the test. By analyzing our results and observing our experiment it was found to that higher values of depth-x, caused our greater offset values for our y and z coordinates.

From our testing, the arm performed relatively close to our expectations. We ran into an issue where the arm would hit the nose before moving to the lips but that was corrected through alteration of the code. We also noticed that the arm interfered with the camera's field of view in between loops for head movement, so we programmed the arm to move to the left of the camera's field of view before taking the next position photo for the following loops.

Accuracy Test: Actual = camera coordinates, Expected = robot coordinates

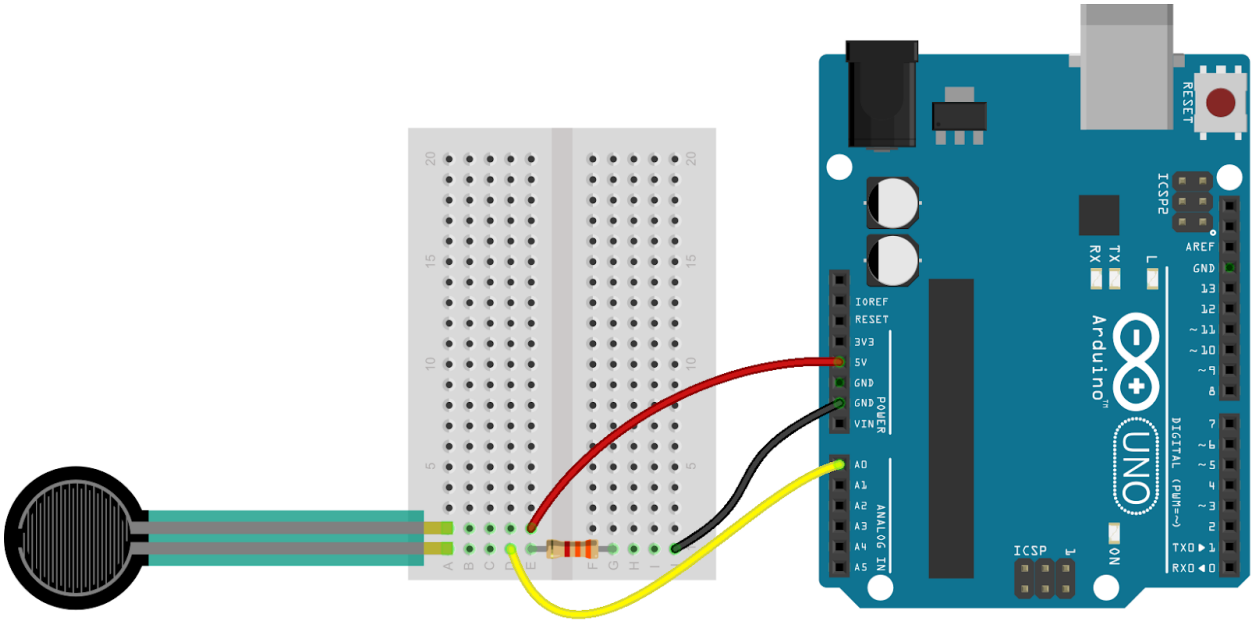
Test#	Actual x (mm)	Expected x (mm)	Actual y (mm)	Expected y (mm)	Actual z (mm)	Expected z (mm)
1	330	321	14	6	3	14
2	320	328	132	154	5	11
3	220	208	61	60	20	14
4	280	266	34	49	9	12
5	260	244	123	109	15	14

Average offset in y (mm)	**12.50**
Average offset in z (mm)	**14.00**
Average offset in x (mm)	**11.80**

To gain a further understanding of the amount of force distributed on the lips, we used a force sensor and ran the system through its normal path. The force sensor was positioned on the lips by hand because taping it on the curvature of the mannequin produced unreliable readings. From the averages of the force readings given in table 5.1.0 and replicating it by manually loading the force sensor, we determined that the force applied was sufficient for application yet light enough for user comfort. Through this experiment we were able to verify that the robotic arm is able to apply the required force, as the specifications claim that the robotic arm is able to withstand a payload of 500 grams or 4.9 N. From our force sensor data we found that the average force for lip balm application was around 1 N based on Table 5.1.0.

Table 5.1.0 Force sensor results (N)

Trial - 1 Force (N)	**Trial - 2 Force (N)**	**Trial - 3 Force (N)**
0.66	0.77	1.2
0.47	1.29	0.77
1.66	1.57	0.81
1.77	0.67	0.62
1.19	0.78	1.3
0.16	1.05	0.62
Average: 0.99 (N)	**Average: 1.02 (N)**	**Average: 0.88 (N)**



Interlink 402 model. The 1/2" diameter

7) Future Alterations

From the test results obtained in section 5 of this report, there is not much about our design that we feel the need to change. The force application and accuracy of the application are what we desired but the movement detection could be improved. Looking at our apparatus from a different lens however, there are many things that need to be done in order to turn this into a product that can be more appealing in a competitive market.

First and foremost it would be ideal to expand our code to be able to apply makeup not to just the lips, but to the cheeks and around the eyes in addition to the forehead. This would require far more than code alteration however, as the medium will change to brushes for blush application and shading work for around the skin. Pencils or fine brushes would be desirable to eyelash alteration or for the skin around the eyes. This would introduce a higher safety concentration and force testing would have to be much more stringent and precise due to the sensitivity of the skin at different areas of the face. In addition, different mediums would require different force applications and different locations for that same medium may require

lighter or harder forces for visibility. All of these factors will need to be considered and implemented correctly and thus will require significantly more time.

Accuracy improvements would be critical as well, since working around the eyes can be extremely dangerous if the eyes are punctured or even touched lightly. As a product this design would be a liability to any company that decides to sell it, if the accuracy is not refined as much as humanly possible. This will require a more refined coding infrastructure and more detailed/strenuous testing.

Aside from increased functionality or arm features, given time we would like to change this system from a computer operated system to a solo user experience. To do this, we will need to develop a graphical user interface and have it on a tablet, or possibly have the GUI integrated with simple press button switches. The controller we purchased has an OLED screen which could be used to display messages to the user. If we decide to expand the functionality however, an application on a tablet or a phone would be more applicable due to the wide amount of options for cosmetic modifications that could be made. Overall, we would like to have a more user friendly and solo experience that is more refined and versatile.

A final modification that we would like to make would be to the base itself. For this apparatus to become a product that could be sold in a market, we would like the base to have a more visually pleasing look. That would mean that the base material be changed to a higher grade plastic or different material altogether. We would also like to have a mirror built into the base pieces, so the user does not have to look elsewhere to see what has been done to them. Tighter tolerances between the fitting parts would also be ideal so that there is much less potential for accidental impacts to have an effect on the base pieces with respect to each other. Overall, there is a lot of work that is needed is this design is to become a fully attractive product in the ever growing cosmetic industry. Given enough time, a proper budget and sufficient resources we believe this one-of-a-kind idea can be further developed.

References

[1] https://cdn.sparkfun.com/assets/9/8/a/c/0/uArm-Swift-Specifications-en.pdf

[2] https://www.ufactory.cc/#/en/

[3] https://www.youtube.com/watch?v=5DnKot3mMSc - Designing Robot Manipulator Algorithms Matlab

[4] https://www.youtube.com/watch?v=q4mIAX-Rcmg Robot safety control and response to unexpected object