**Shilpa Mitra: Redefining Quality Standards for Toy and Handicraft Makers with 3D Scanning**

**ii) Abstract**

**Objectives**

Shilpa Mitra was created with the objective of an affordable 3D scanning system under ₹ 2000. It is paired with an app that uses images and rough 3D scans to allow small—to medium-scale industries to automate their quality control. This enables higher-quality produce while also removing the room for human error.

**Beneficiaries**

The primary beneficiaries of this solution are small to medium-scale toy/handicraft makers, these people face a unique issue of having enough quantity for manual Quality control to not be viable but also do not have enough disposable capital to implement an automatic solution.

**Value of Results (Usage)**

The given system uses the power of 3D scanning and image processing to seamlessly automate a business's Quality control line, allowing manufacturers to maintain and regulate a higher quality of product allowing them to regain market share in the higher income brackets of India.

**iii) Background**

With India’s handicraft and toy industry rapidly growing and currently employing around 7 million people, the need for Quality control is essential for not only creating an export line for foreign trade but also maintaining local presence with the influx of cheap Chinese toys entering the market.

The current issues Indian manufacturers ( especially small-medium scale manufacturers) face include the inability to regulate quality among the hundreds of goods they produce having not enough resources to manually inspect each one of them while also not having the funds to maintain Quality Control machines.

**iv) Statement of Problem**

Rural artisans lack of access to modern technologies that could enhance their production efficiency and product quality, making it difficult to compete with mass production while also forming an unhealthy dependence on word of mouth and local retailers.

**v) Research**

**Present Methods of Tackling the Problem:**

Currently, most Small/Medium scale manufacturers rely on eyeballing for quality control, leading to many errors, this can be most noticed in the percentage of defective toy/handcraft imports from around the world with 64% of all toys and handicrafts failing key Quality Control checks during import from their country of origin **[1]**

**Proposed Solution:**

The Shilpa Mitra project aims to address this issue by offering a low-cost 3D scanning system (under ₹2000) paired with a mobile app for small manufacturers. The system captures rough 3D models of products and cross-references them allowing us to get an in-depth understanding of issues like cross-threading of screws, misaligned/cross sectional threading in cloth/fabric, etc all in real-time, This technology therefore helps us ensure consistent quality control while remaining economically feasible.

**Alternate Solutions:**

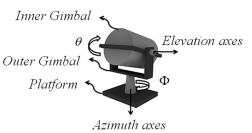
3D scanning technology enabled by LIDAR is currently primarily used in the automobile industry, with such systems able to spot issues prematurely saving rework costs by up to 40% and improving labor efficiency by up to 15% by such automation. However, these systems are financially out of reach for almost all manufacturers only being used by General Motors and Lexus **[2] [3]**

**Novelty of Approach:**

The novelty of Shilpa Mitra lies in its cost-effectiveness primarily achieved by leveraging affordable hardware, the project provides small-scale manufacturers with a solution that typically would have been too expensive. Allowing them to eradicate their reliance on manual inspections, which are prone to human error while also being time intensive, the proposed system offers more reliable and precise quality control for its price while mirroring the benefits seen in larger industries—such as improved accuracy, reduced labor costs, and faster defect detection—at a fraction of the cost.

**vi) Technical Report**

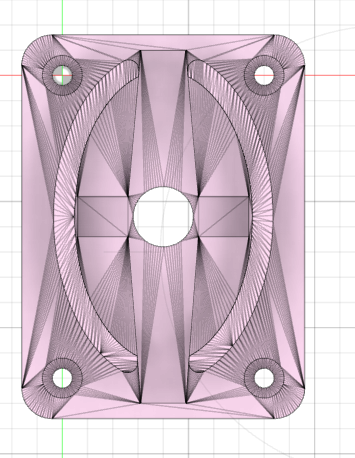
The primary working concept of our project is using the axels of a servo motor and a custom 3d printed mount to angle and orient a LIDAR sensor at a given spherical coordinate, using the distance obtained from the lidar letting it be the spherical distance( alpha).



In this case, they are letting the angle of the servo controlling the angle Φ be

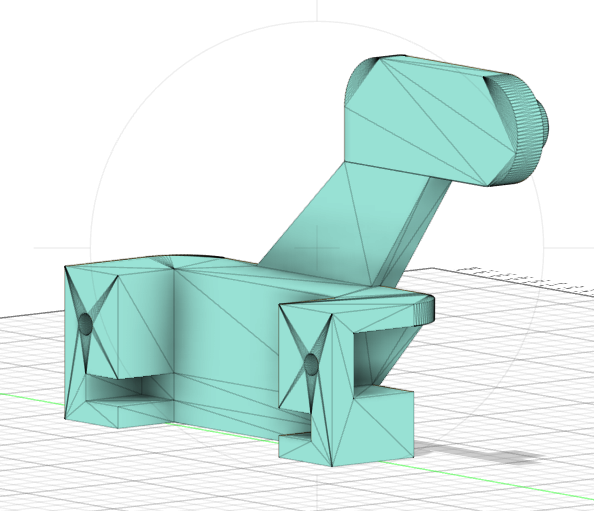
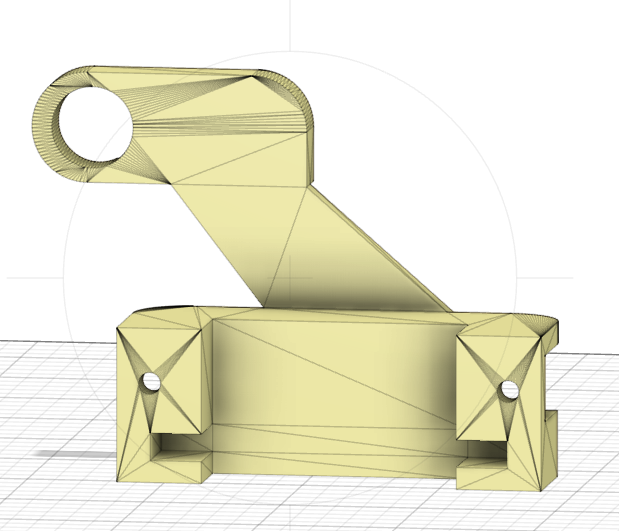
And the angle of the servo controlling Ө be . ( image credit **[4]**)

The hardware part of our project relies on a 3d element custom-designed for our application along with the necessary control electronics, all of these are showcased in the following pictures





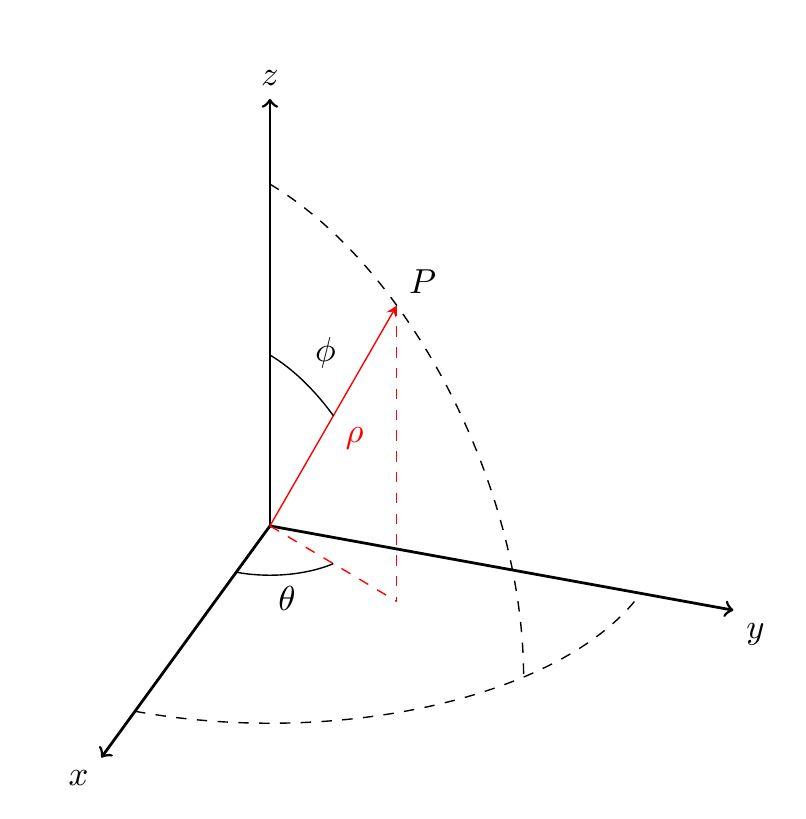
X-axis Servo mount Y-axis Servo base



y-axis servo mount X-axis servo support

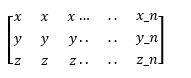
All of these pieces can be attached from readily available parts helping in its accessibility, the control electronics include a microcontroller, two servos and capacitors.

Moving on, using the aforementioned and plotting such coordinates in the spherical coordinate system we get this diagram which can be used to derive the cartesian coordinates.

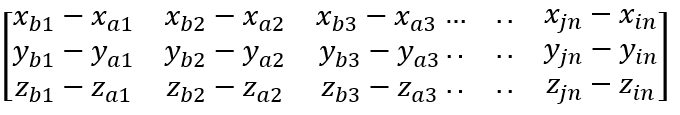
In this case, breaking this picture into three right-angle triangles and using some trigonometry for our specific scenario we get the following equations. 

)

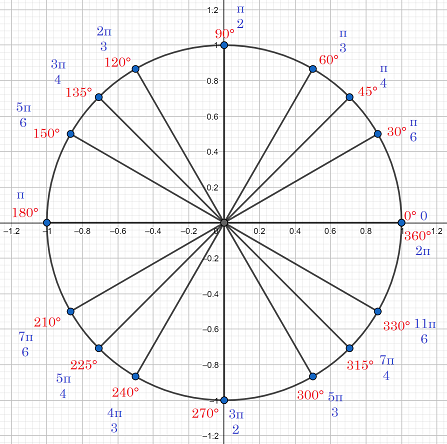
With these readings becoming essential at their n’th respective reading of .

’G‘ has been assigned to be a surface with different coordinates starting from moving from π to 0 with being constant whereas in the next iteration alternates from π to 0 while staying constant .

∴

Surface ‘G’ is then computed in quadrants referenced by computing the net differences between the reference and to be-tested surface computed by the density of values in a given range represented by

=



Surface ‘G’ is therefore computed in quadrants referenced by the given picture.

The density of the values in terms of concentration has been computed and is used with the normal of the coordinate. Hence the known X, Y, and Z coordinates are plugged into the following equation.

while

To compute the depth we need to establish some of the prerequisite variables which include important constants like the total difference in a given computation, given by this equation.

Total difference = (1)

This integral (1) can be evaluated by the method of reinmen summation to give us a final output of the

Total difference

while the gradient or the rate of change can be represented by the following equations -

Gradient w.r.t =

Gradient w.r.t =

Furthermore, to find out the angle of a specific point on the surface, a perpendicular vector on the side of that particular point (also called the René gradient-based surface orientation) is used.

So for each point on the surface G, we can assign the perpendicular vector ‘N’ which can be described as follows:

)

[ The ‘-1’ ensures that the vector the perpendicular to the given point]

By using the relation between dot products and perpendiculars from coordinate geometry **[7]** , it can be concluded that

(2)

in which N is the defined perpendicular, K is a unit axis vector and N K is the dot product of the two whereas would be the magnitude of the unit vector with being the magnitude of the perpendicular.

now from (2)

we can evaluate

= 1

and also evaluate =

= 1

whereas is much harder to compute with the magnitude being equivalent to

Using these simplified terms we can evaluate the final value of (2) to be such

giving us the final equation

The angle modifies the inherent depth of the camera by giving us an understanding of the tilt of an object with respect to the axis, so, for example, a higher surface ( is higher ) would indicate a lower actual depth, causing the apparent depth to be higher as it is inversely proportional to the displacement which varies across the length of the angle giving us a perspective on the depth **[8]**.

This pattern can be used primarily to detect important quality constraints like cross-weaving of fabric, as it typically leaves a periodic variation in surface height around the axis of weaving it can be seen below on our implementation of this segment through our c# code as shown in the next page

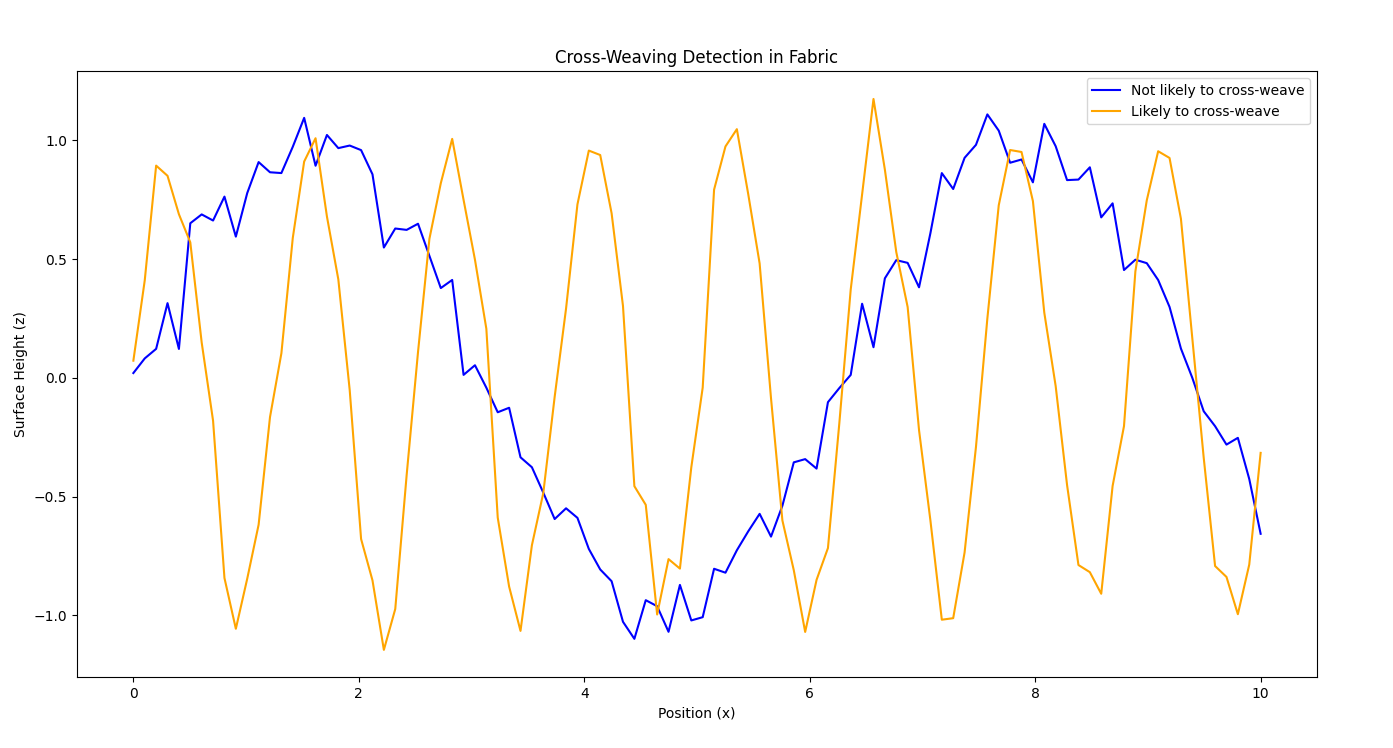


This pattern can be used primarily to detect important quality constraints like cross-weaving of fabric, as it typically leaves a periodic variation in surface height around the axis of weaving

∴ we can create a 3D height map of sorts which then allows us to calculate the surface normal which can be used to form a threshold, with this variable, you can compute the Laplacian 2nd derivative for the following terms

, if these terms show alternative or periodic patterns, it is highly likely that, that particular portion has undergown cross weaving, this area can then be flagged to be cross-referenced to the input picture to confirm with this being better represented in c# as follows 

This for example is how the computation would look like for known variables :

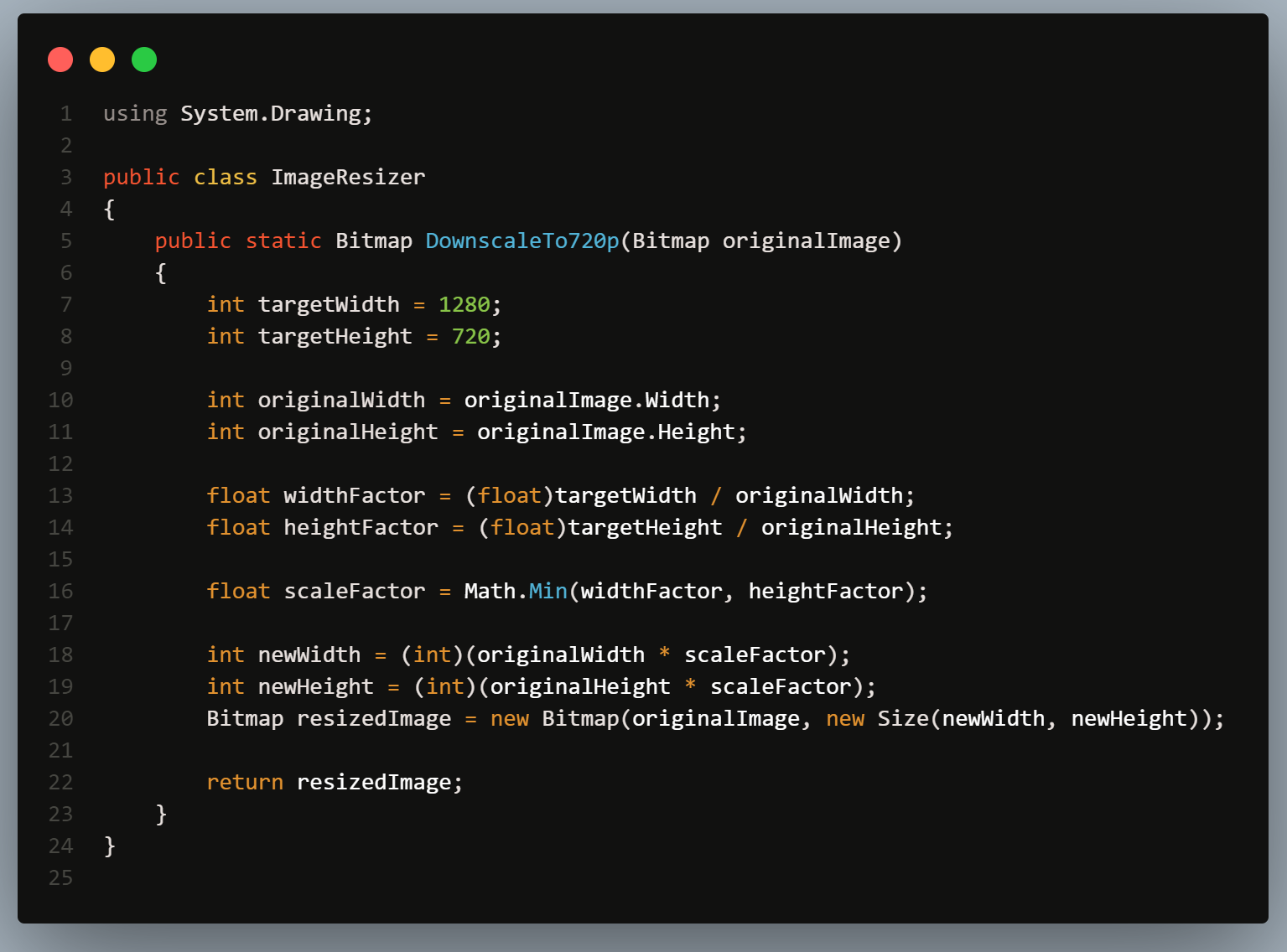
Now, moving on to the second technical element of our project, the image processing element using pixel height and pixel ratio algorithms integrated with an AI model trained on that specific product to enable us to further determine quantities like pixel depth and so on however an important constraint in our model would be differing pixel ratios from different images from different cameras and so on, therefore we use a resolution standardisation algorithm to ensure consistency, in this case, we are using the below parameters for our system to downscale images to our desired 720P resolution for its common accessibility and it being a highly composite ratio allowing pixel compression algorithms to run more efficiently.

720P in terms of pixel length and width can be represented as 1280x720 pixels respectively, to now form an efficient compression system we need something known as the scaling factor, denoted by :

for the width

for the height

these factors can be multiplied with the original number to find the pixel per compression ratio to do so, an example implementation of this would look something like this in C#,



now to calculate pixel heights and angles for cross-referencing, we employ a common pinhole camera due the wide availability and choice in our required compression algorithms **[5].**

This model in particular relates 3D points in space to the 2D points in the image with it being denoted by

where :

is the 3D point in the world.

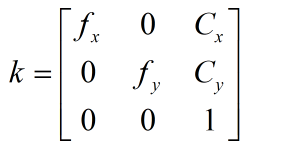
is the corresponding point on the 2D image plane.

is the scaling factor due to depth.

**𝐾** is the intrinsic camera matrix (accounts for focal length, pixel scaling, and principal point offset).

**[𝑅∣𝑡]** represents the extrinsic parameters (rotation and translation matrices) that transform 3D world coordinates to the camera's coordinate system.

The intrinsic camera matrix ‘K’ Being essential , in this case it can be represented as:



where and are the focal lengths along the x and y axes and , represent the coordinates of the principal point (the point where the optical axis intersects the image plane).

Now from **[6]** and the matrix ‘K’ we can derive the height of each pixel represented by

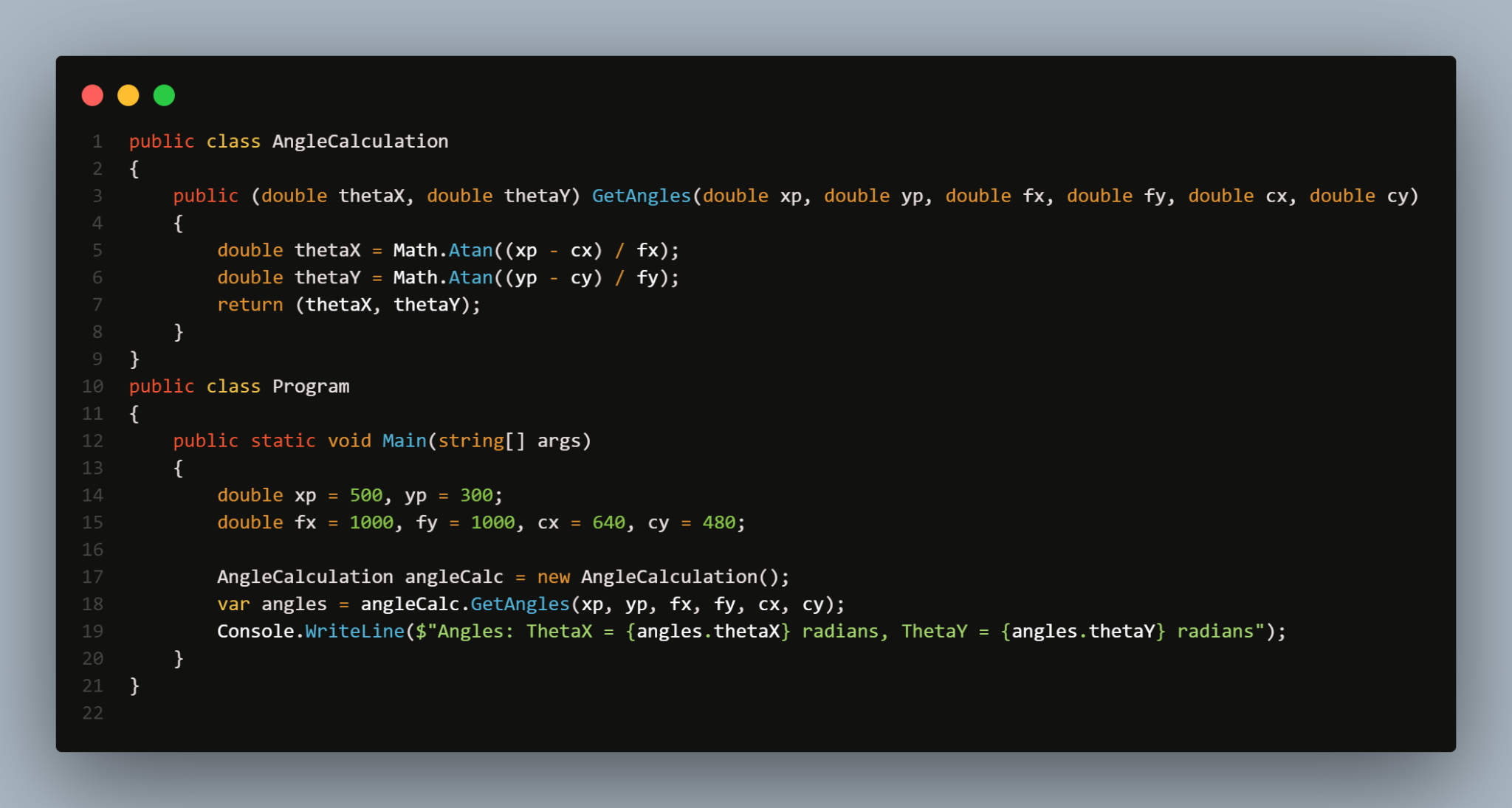
where is the depth and is the real world height derived from the 3d scan.

This simplistic and pre-verified approach allows to program an efficient and reliable algorithm like the one given below with ease :

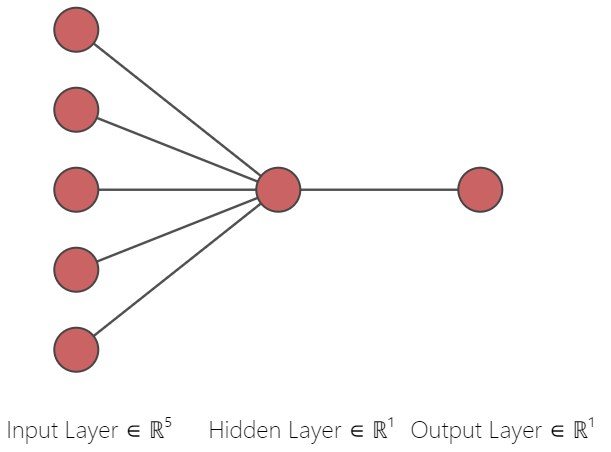


Now furthermore we integrate our approach with the projective geometry method **[9]** to the derive the angle from pixel lengths to be represented by the following equations

in which (u,v) represents the coordinates of the object in the image, this also can similarly be integrated in c# with the following code :



All of this data is stored in a separate array for use at a different time after which the image is redirected to the cross-reference algorithm prepared in Python, it particularly uses tensor flow to form relationships between all the different variables obtained throughout this paper with reference to this particular neural network :



In the diagram, the input layer would be the 5 images given to us by the user with the hidden layer being all the data we acquire from the 3D scanning algorithm which will then be interpreted to form the output layer which is in the form of a similarity score determined by a variant of the following algorithm.



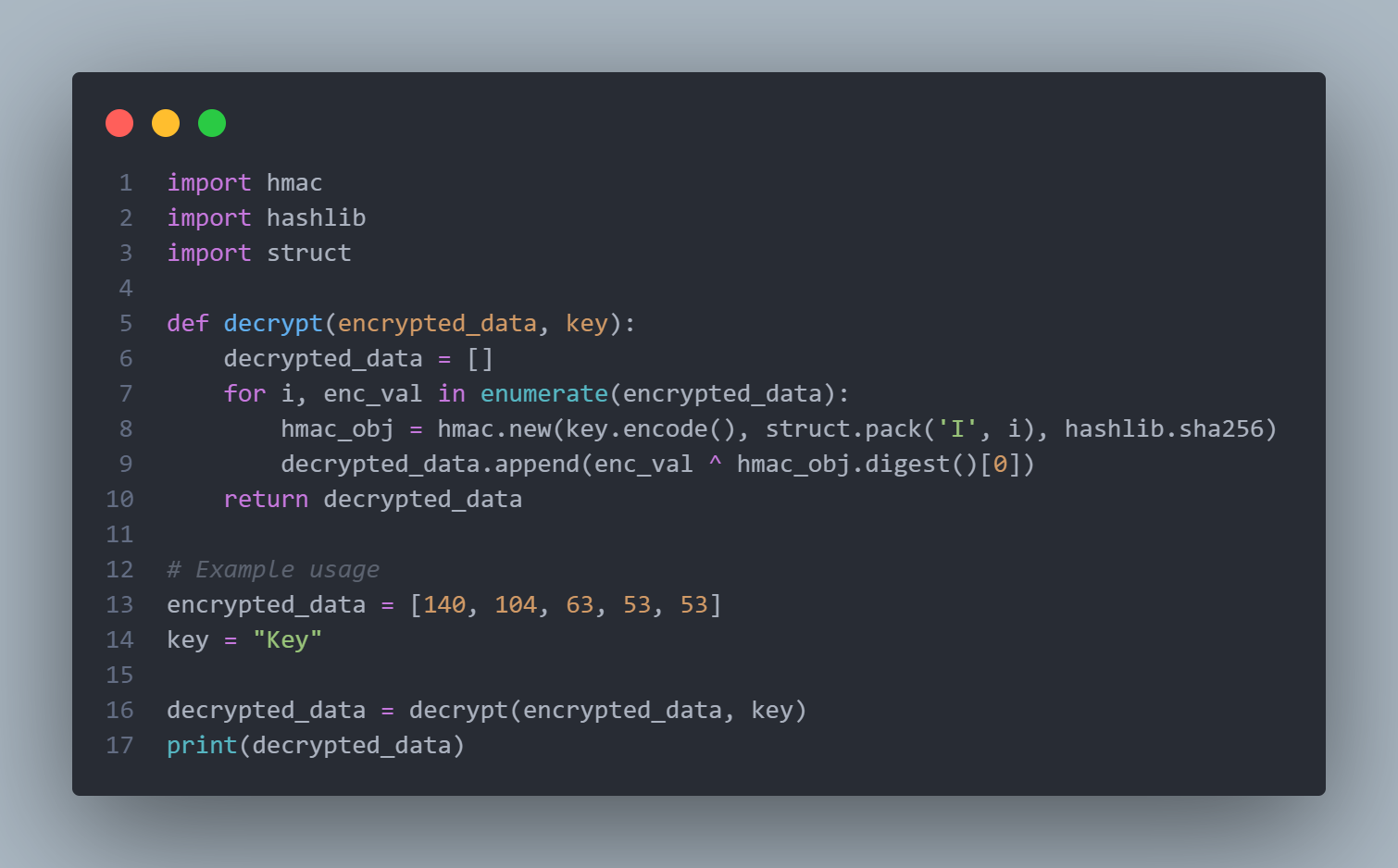
all of this is possible due to our communication between the hardware and software using the I2C protocol along with an encryption set, tailor-made for our approach in which the c# part is the transmitter with it being attached to hardware and handling most of the computations were as the python part is

receiver being the helm of front end computation, below is our implementation

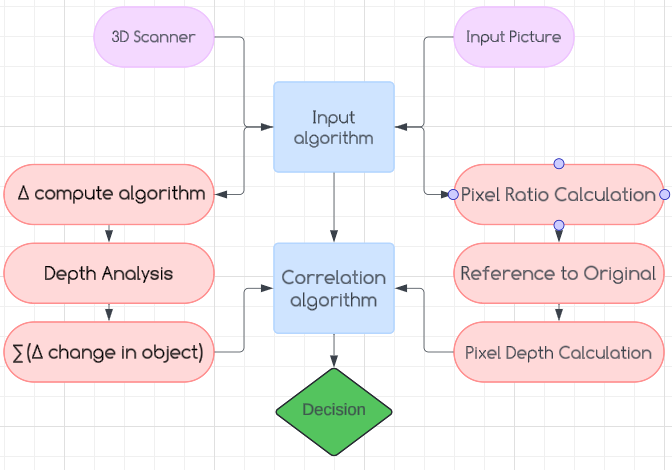
**Sender (C#)**

****

**Receiver (python)**



All of this computation is managed by the feedback loop as shown below on the client side of the server



**vii. Results**

In order to better test our hypothesis, we’ve done several tests on primarily toys and pieces of handicraft clothing, below are two pictures of our test subjects :

[ Fig 1 ] and [ Fig 2 ] respectively



The following sample is a comparison of [ Fig 1 ] before and after some potential damage :

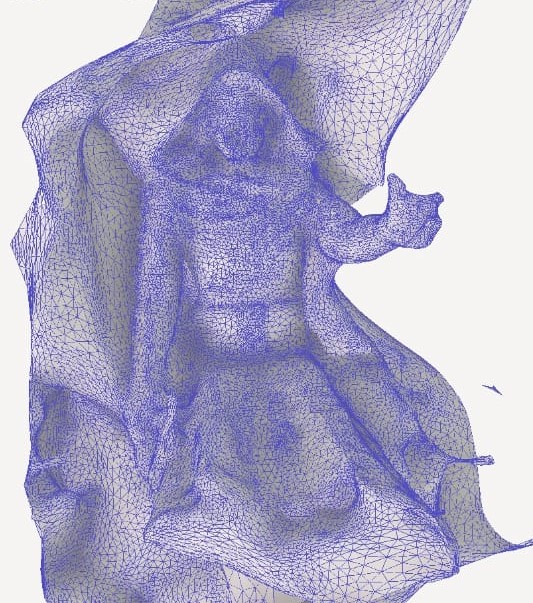
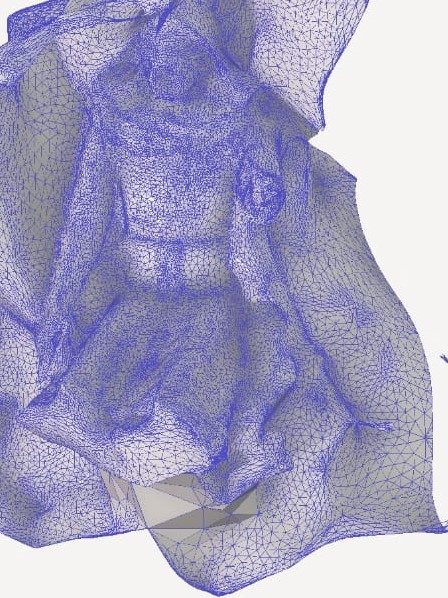


and this is a similar comparison between [ Fig 2 ] in a regular and torn state:



Now contrasting this with our 3D scans gives us some interesting insight, the Toy figurine was placed under ample light, and took around 20 minutes to complete a full scan with

32400 readings across the entire surface, after using Sccipy’s inbuilt filtration algorithm and using the matplotlib library to form a cloud mesh here are our results :



before cut After cut

although not noticeable , while zooming in on the area of damage a significant decrease in Z axis height is observed indicating a greater depth in the surface implying a cut of sort’s which is accurate as seen from this picture [ Fig 3 ].



[ Fig 3 ]