Global Scientific JOURNALS

GSJ: Volume y, Issue o, a l·ŇŎK 20нл, Online: ISSN нонл-фмус
www.globalscientificjournal.com

3D MODEL GENERATION OF REAL-OBJECTS USING 2D INFORMATION

Tonni Das Jui, Md. Alium Basir, Md. Ashraful Alam

KeyWords
2D picture, 3D model, Abstraction, Color-information, Depth-information, Extraction, Real-objects, and RGB.

ABSTRACT

The process demonstrates a method of generating a 3D model using depth and color information of real-object using our proposed algorithm. The primary idea of the method is to manipulate extracting of RGB and depth data from the 2D colored picture of that object and abstracting them within arrays to plot in three-dimensional frames to generate the 3D colored model of the real-object. The proposed system comprises of 3 phases. The phases include acquisition of color and depth information of real objects, extraction, and abstraction of RGB and depth data, generating a 3D model using our proposed algorithm method. Whereas, our proposed algorithm focuses on creating several 1D arrays for depth as well as the color data meaning Red, Green, Blue pixel values separately. By mapping the depth values accordingly pixel-wise 3D black and white models can be created where the background is also included. By limiting the depth values according to the position of the object the background can also be removed to construct the focused object's 3D model. Furthermore, our algorithm maps the RGB arrays in the black and white models to add proper colors pixel-wise. As a result, the colorful RGB 3D model can be achieved. Lastly, we implement or approach on three different case scenarios to evaluate the efficiency of the approach.

Introduction

For the amazing presentation and visualization of 3D models, the implementation of them are getting popular day by day among professionals and common people in every possible aspects. Designers who work in this area are continuously researching and working to improve previous methods and invent efficient newer methods to implement 3D models which are quite praiseworthy. From starting to comparatively less serious fields such as animation, gaming, and films industries to serious fields for example biological and medical science, this idea of 3D models is being widely used. However, to create 3D models from scratch using basic software's like blender and sketch is very time consuming and also need a great amount of effort and cost to give it a definition that is closer to reality. Due to this, we got motivation to work in this field for our thesis to contribute and propose a system or method to implement 3D models more easily which can generate models within a short amount of time.

This research is done by taking images by kinect sensor XBOX One just from one angle, to create 3D models. The pixel quality of visual representation of the 2D images is viable to implement as inputs for any algorithm. Moreover, due to its increasing popularity, console gamers mostly have these Kinect sensors for playing. This thesis may also help in the field of animation and have more serious implementation in the field of medical science.

In order to implement our method easily we researched and studied various available methods of generation of 3D models. In our proposed approach, we created 3D models from the color and depth data values acquired by Kinect Xbox-ONE. The research work is on creating 3D models cost effectively and implementing proposed algorithm with maximum efficiency, main objectives are:

- Extract depth and color information of 2D image of the object
- Use one dimensional array in order to avoid the complexity of manipulating three dimensional matrixes to abstract huge data
- Improve model quality in different scenarios

Besides, Rest of the paper is organized to describe object cases which ensure successful efficiency of our proposed method.

Literature Review

With the rapidly growing technological based industries, 3D model creation and its implementation in various sphere of life has been increasing in an enlarged scale. To create a 3D model of an object, the depth data and color data is needed at the very beginning. The demands of using 3D model vary in different industries as well as in case of applications according to need. In gaming world, Video segmentation is an essential building hinders for extra ordinary state applications, for example, scene comprehension and communication investigation. While exceptional outcomes are accomplished in this field by best in class learning and model based techniques, they are confined to specific kinds of scenes or require a lot of clarified preparing information to accomplish protest division in nonspecific scenes. Then again, RGB information, generally accessible with the presentation of purchaser profundity sensors, gives genuine world 3D geometry contrasted with 2D pictures [7]. A constant framework for programmed formation of 3D confronts models from a video succession is introduced in another study of framework for programmed production of 3D confronts models from a video arrangement framework for programmed production of 3D confronts models from a video arrangement[6]. The framework comprises of a novel plan to extricate confront shape and a 3D show adjustment technique in light of a minimum square approach. The proposed confront shape extractor, which is exact and computationally shabby, depends on an oval controlled by three grapple focuses. The minimum square approach is chosen to adjust a non-specific 3D model to include confront, limiting mistakes between highlight focuses on a made 3D confront and separated component focuses from input video. Test comes about demonstrate that the proposed framework can productively manufactures 3D confront models, which are good with MPEG-4 facial items, from a video arrangement with no client mediation [3]. Another method of processing 3D model can be done in CAD/CAM system which is capable of modifying, archiving as well. The challenging part is the transformation procedure where data need to acquire from paper drawings [2]. Again, 3D imaging methods, generation of 3D models and illustration of 3D points in cloud at different positions along with their applications and implementations are by discussed by the authors in [4]. A detailed assessment has been given by the authors about Kinect for Windows V2 Sensor for the purpose of reconstructing 3D models in [1]. In [5], the authors provided a comprehensive record of the variants (that are different from each other) of closest point algorithm which could be useful for the arrangement of various 3D models for the same object into a single position.
A large portion of 3D demonstrating devices' working interface is intricate. In [6], a quick free outlining 3D demonstrating strategy with edge understanding drew nearer. Free outlining is this present technique's information. A 3D demonstration was built in a brief span. Some self-convergence chart can be perceived however include some edge record data in this strategy.

Proposed Methodology

Our proposed method's first phase is about acquisition of RGB and depth images of the object using a device that encompasses depth and color camera while capturing an objects visual.

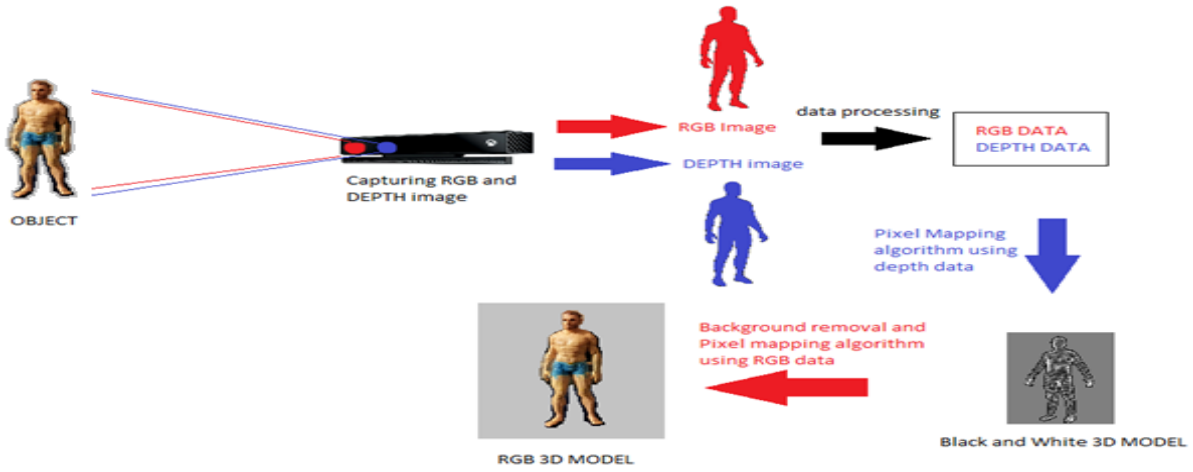

Fig. 3.1: System Architecture.

For collecting and arranging the massive information of RGB and depth images Matlab is used as numerical computing environment. The second phase focuses mainly on extraction of the data of color pixels and depth pixels and abstracting them in a suitable form to be implemented as input variables on the algorithm to create three dimensional models as the output. The final phase refers to 3D

model creation by implementing our proposed algorithm in reference to pixel mapping algorithm. In our approach, we used Kinect 2.0 sensor to create the colored 3D model for our research purpose. Any device that provides with information of RGB colored picture and depth picture can be used for implementing our proposed methodology to generate 3D model. Our proposed method's third phase emphasizes on our created algorithm which is based on pixel mapping technique. The algorithm is based on 1D array of mainly 4 parameters which is acquired from the 2D array data. These 2D array data's are extracted from the images taken by Kinect sensor. The 4 parameters we use in the process are X, Y, Z and RGB.

Step 1: Acquisition of RGB and depth information of real object: For getting RGB and Depth information, we capture images

Fig. 3.2: RGB image example.

Fig. 3.3: Depth image example.

the object using our preferred device which is Kinect sensor with connecting it to a multi-paradigm numerical computing environment and proprietary programming language. MathWorks developed MATLAB which allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages. Our initial plan is to use Matlab for processing the images (depth and color) we collect via capturing picture of the object using kinect sensor. Matlab provide us with several options in the image acquisition section, referring to two options that we will use for acquisition purpose. One is for taking RGB image and other one for depth image. In our case, by fixing the RGB picture resolution into 2000*1000 (approximately), picture exportation is initiated and image is stored in windows bitmap format to move to the next phase which is extracting enormous data for abstraction. By following same process we captured the depth Image and stored in TIF format. Here for our implementation purpose, we chose 640*480 resolutions size for the depth image. The first object we chose and implemented our algorithm was a rubrics cube and the medium size resolution was best fit for that object (Photo was captured from approximately 10 meters). For processing data in the next step we need to resize the RGB image to 640*480 resolution as our pixel mapping algorithm requires both image data (RGB and depth) to be same resolution.

Step 2: Extraction and abstraction of RGB and depth data: We generate the RGB and Depth images information from phase one. We import those images in our Matlab for extracting the RGB and depth data. For extracting the data, we need to import the images in Matlab. In our case, we stored RGB images as Unit8 format provided by Matlab which is 2D array containing colored pixel's information of each color in reference to red, green and blue values in this arrangement of a [3*1] matrix representing red, green and blue color pixel value

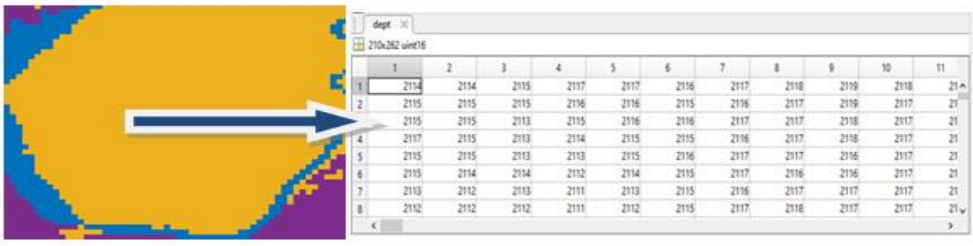

Fig. 3.4: Data extraction

We use a function to make 3 separate 2D arrays for RED, GREEN and BLUE from the previous array. We take depth image as Unit16 format which is a 2D array containing distance values from Kinect sensor, pixel wise. So, from here we accumulate four 2D arrays which need to be used in our proposed algorithm in the next step.

Step 3: Generating 3D model using the proposed algorithm: To create the model proposed algorithm needs to be used and the inputs need to be arranged in an appropriate way. To demonstrate the whole system our proposed method refers to dividing the

whole process into two segments.

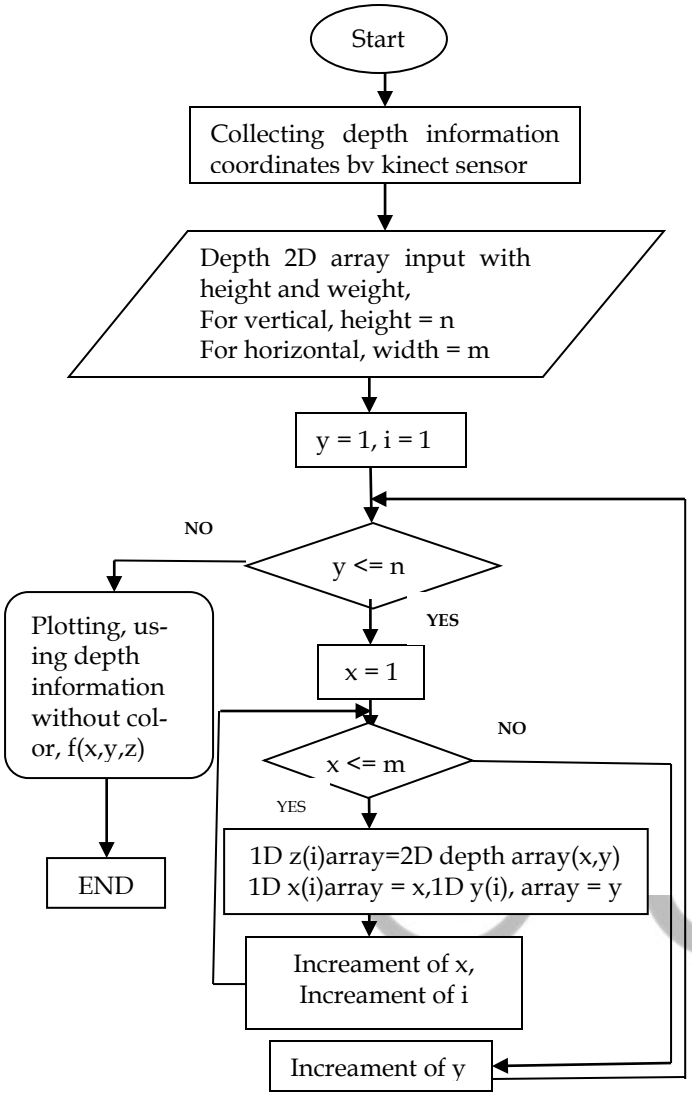

Fig. 3.5: Flow chart of the algorithm for the 3D model generation without color.

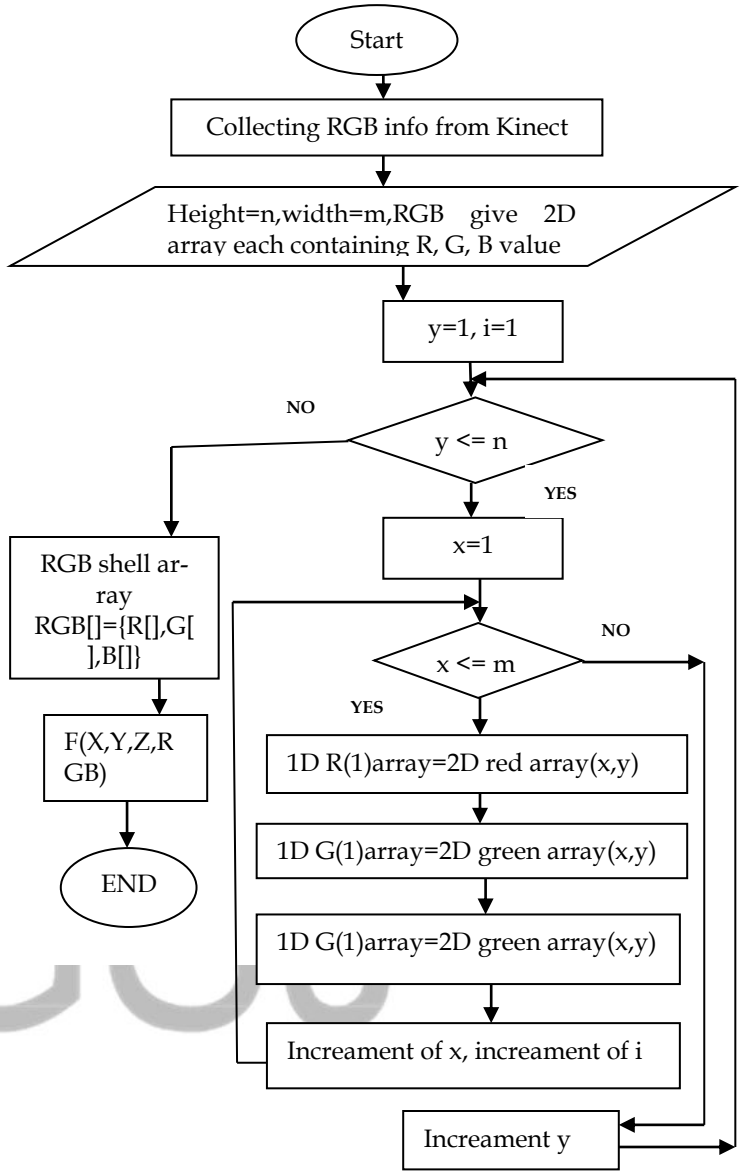

Fig. 3.6: Flow chart of the algorithm for adding RGB color.

First the depth input should be arranged. Depth information's of the Kinect sensor are in 2D form. So in the first Fig. 3.4 height and width of the array can be replaced with m and n temporary variable. In our flowchart in Fig. 3.5 vertical and horizontal array places are filled with depth values. So for the 1st value that should be plotted, it is in the horizontal and vertical 2D matrices first index of the array list shown as Table. 3.1. Another 1D array is created to transfer the 2D metric's depth values in a way so that when the first horizons last value is transferred, 2D metric's next horizons first value is moved in the 1D array's next index shown as Table. 3.2.

Width →

↑ Height

DepthOf(1,1)	DepthOf(2,1)	DepthOf(3,1)	DepthOf(4,1)	DepthOf(5,1)
DepthOf(1,2)	DepthOf(2,2)	DepthOf(3,2)	DepthOf(4,2)	DepthOf(5,2)
DepthOf(1,3)	DepthOf(2,3)	DepthOf(3,3)	DepthOf(4,3)	DepthOf(5,3)
DepthOf(1,4)	DepthOf(2,4)	DepthOf(3,4)	DepthOf(4,4).	DepthOf(5,4)
…….	…….	…….	…….	…….

Table. 3.1 Two dimensional array of depth information

DepthOf(1,1)	……	DepthOf(5,1)	DepthOf(1,2)	…….	DepthOf(4,4)	DepthOf(5,4)	…

Table. 3.2 One dimensional array of depth information that has been constructed.

This way all the depth values (Z axis) are transferred in the 1D array. For plotting 3D model, there should be other two dimensions too (X axis and Y axis). For X axis's values and Y axis's values, other two 1D arrays are to be constructed. These arrays system are simpler than z-axis's data table construction. For both the x-axis's one dimensional array and y-axis's two dimensional array, for the 1st value that should be placed, it is in the horizontal and vertical 2D matrices first index number of the array list. Another 1D array should be created to transfer the 2D metric's depth values in a way so that when the first horizons last value is transferred, 2D metric's next horizons first index number should be moved in the 1D array's next index . Than the plot functions with X, Y, Z [1D arrays] values will plot a colorless 3dimentional model (it will plot height*width times, as each time it creates one pixel per X, Y, Z co-ordinate plane. So creating height*width times will draw the whole pixels in one X, Y, Z co-ordinate plane).

The second stage demonstrates for adding the color. After doing all instructions described in stage one flowchart shown as Fig. 3.5. Flowchart shown in Fig. 3.6 is the complete algorithm to add color pixels in appropriate position in reference to the depth pixel values.

To implement this method, introducing with shell array idea is needed. Shell array can hold shell values in one index of the array. One shell can hold multiple values in any sequence (for example, shell Array [1] = {10, 40, 20}).

Width →

Height ↑

R&G&B(1,1)	R&G&B(2,1)	R&G&B(3,1)	R&G&B(4,1)	R&G&B(5,1)
R&G&B(1,2)	R&G&B(2,2)	R&G&B(3,2)	R&G&B(4,2)	R&G&B(5,2)
R&G&B(1,3)	R&G&B(2,3)	R&G&B(3,3)	R&G&B(4,3)	R&G&B(5,3)
R&G&B(1,4)	R&G&B(2,4)	R&G&B(3,4)	R&G&B(4,4)	R&G&B(5,4)
……	……	……	……	……

Table. 3.3: Two dimensional array of color information (red and green and blue values).

First we extract the red color values in a two dimensional array and then accordingly other colors (green and blue color values) are also to be moved to separate two dimensional arrays.

Width →

Height ↑

Red(1,1)	Red (2,1)	Red (3,1)	Red (4,1)	Red (5,1)
Red (1,2)	Red (2,2)	Red(3,2)	Red (4,2)	Red (5,2)
Red (1,3)	Red (2,3)	Red (3,3)	Red (4,3)	Red (5,3)
Red (1,4)	Red (2,4)	Red (3,4)	R&G&B(4,4)	Red (5,4)
……	……	……	……	……

Table. 3.4: Two dimensional array of color information (only for red pixel values).

From the extracted red colored data in different 2D array, along with other colored data in their separate allocated 2D array, total 3 sets of 2D array is constructed. Now, the process of one dimensional array construction can be proceeding (without separating the RGB values individually, our algorithm could not be implemented. For our algorithm 1D array is an essential concentration.). Table. 3.4 shows the 1D array where the 6th index contains Red (5, 1) which is the 2nd rows 1st index's content of 2D array.

Red(1,1)	……	Red (5,1)	Red (1,2)	…….	Red (4,4)	Red f(5,4)	…..

Table. 3.5: One dimensional array of color information (only for red values)

Finally from the 3 sets of 1D array each containing red, green, blue color's values of every pixel, our algorithm now focuses on making a shell array for adding color to the pre-described flowchart Fig. 3.5, sequentially shown in Fig. 3.6. The first index of the shell array holds 3 sets of values of the (1, 1, 1) co-ordinate and carries constructing the shell array till n (=height*width) times accordingly.

ShellArray({r(1),g(1),b(1) }	……	ShellArray({r(m),g(m),b(m) }	ShellArray({r(m+1),g(m+1), b(m+1) }	…….	ShellArray({r(n-1),g(n-1),b(n-1) }	ShellArray({r(n),g(n),b(n) }

Table. 3.6: One dimensional array of color information (for red, green, blue values).

Now, the shell array provides with all the essential information to add color to the plotted model. Generally these 2 stages of plotting the 3D model methods are parallel process, but for better understanding of the 2 types of data handling systems to implement our algorithm, these are explained in two stages.

Model generation in three cases

Although a conclusion may review the main points of the paper, do not replicate the abstract as the conclusion. A conclusion might elaborate on the importance of the work or suggest applications and extensions. Authors are strongly encouraged not to call out multiple figures or tables in the conclusion—these should be referenced in the body of the paper.

4.1 Implementation (3Dimensional model Generation for three objects): We used human as an object overall to implement our proposed method. The focused objects are,

4.1.1 Case: picture of the human figure (Including background without color)
4.1.2 Case: picture of the human form: 3D model Generation (excluding background without color)
4.1.3 Case: picture of the human form: 3D model Generation (excluding background of Object with color

The case involves a human as the object of which we would like to generate a 3D model followed by brief explanation of every step implementation. These steps are selected according to implement our method in various ways to ensure its credibility for generating a 3D model of those objects in any possible scenario.

4.1.1 First case: picture of the human figure (Including background without color): In this case the man's hands are vertically and horizontally parallel. After getting all our data's by acquisition and processing we used the 1D full arrays X, Y and Z which have height*width number of values in each one, to create a 3D plot. X, Y, Z arrays have values in it row wise so the position -i value of each array maps a particular point on the 3D plot according to the real depth image. By plotting all the points we get a 3D representation of the object.

Fig. 4.3: Actual image taken by Kinect sensor.

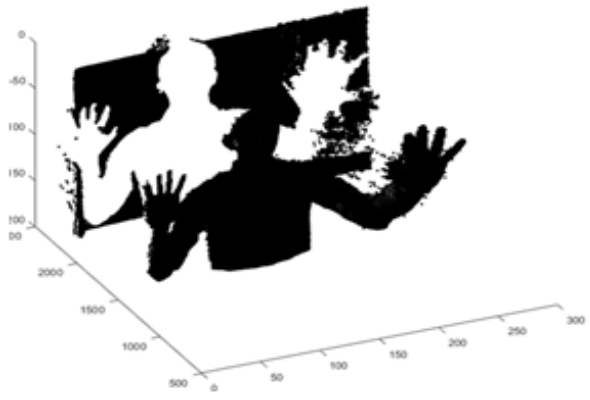

Fig.4.4: Depth image taken by Kinect sensor (left-front view).

X, Y, Z arrays have values in it row wise so the position -i value of each array maps a particular point on the 3D plot according to the real depth image. By plotting all the points we get a 3D representation of the whole depth image shown in figure fig 4.4.As the depth image is not clearly understandable as an image the RGB image is shown in fig 4.3 to clearly visualize the 3D simulation in

black and white.

4.1.2 First case: picture of the human form: 3D model Generation (excluding background without color): The model of the human can be rotated at any angle to better visualize the 3D plot. We have shown here the visualization from a particular angle .From the fig 4.4 we can see there are background plotted along with the object (here the human body). To remove the background we edited the algorithm to filter out the points which has Z values more than the objects last maximum Z values and plotted the object only in 3D plane again shown in fig 4.5.

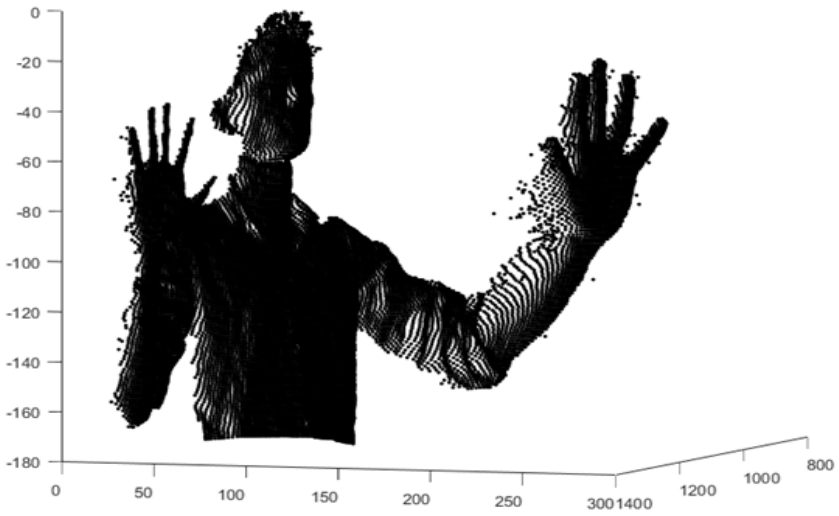

Fig. 4.5: Depth image of actual object (after image processing, front-right view).

4.1.3 First case: picture of the human form: 3D model Generation (excluding background of Object with color): From the fig 4.5 we get the 3D model for the object. Now to add color we had to use our self-made pixel mapping algorithm which gives each point the exact color from the RGB image.

Fig. 4.6: Object's image after applying algorithm (front-right view).

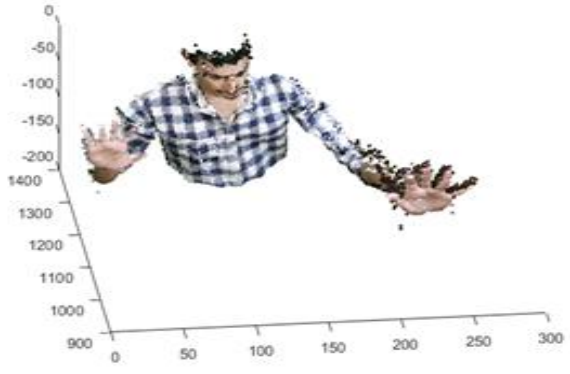

Fig.4.7: Object's image after applying algorithm (Top view).

We took the RED, GREEN and BLUE arrays similarly as the X, Y and Z arrays row by row to perfectly match each point's red, green and blue values. As we needed a full array with the red, green and blue values we had to use a shell array of named RGB which contains merged values of red, green, blue row wise in 1D array. Using this 1D array along with the previous plot we created the 3D

model of the object with exact color in each point. The model can also be rotated for better visualization. We have shown the model From 2 particular angles in fig 4.6 and fig 4.7.

Result

The comparison of a measurement with a known standard is used to determine whether the measurement is reliable. Estimation precision is recognized as the distinction between the estimation of a factor and the acknowledged an incentive for that factor from a trusted outside source, or the rate by which the two esteems vary. We tried to build a method, using some functional values. These values are calculated based on some functions that should be excluded from a standard acknowledged value that is exactly hundred percent or we can also say a whole percent.

So, our proposed method of measuring the accuracy rate depends on some functions. We have found out that these functions should be called Depth and RGB pixel difference, software incompetence, Motion limitations, Kinect sensor XBOX one camera placement etc. The first two functions are theoretically representational. Other functions contribute on accuracy lacking of the model but not predominantly like the first three functions. So these functions are,

Depth and RGB pixel difference: The pixel values of original source depth picture and RGB picture are not the same. It's a major obstacle that is in between creating a perfect model. The depth picture's resolution is always less than the RGB valued pictured resolution. This problem occurs and we actually have to lessen the RGB valued pictures resolution to solve this problem which creates a pixel gap which has its effect on our pixel plotting algorithm. For better explanation our 3.7.1(a) figure's depth picture has a resolution of 56*47 and its RGB picture's resolution was 78*64. Here's their difference is 78-56 * 64-47 = 22 * 17 which is 38% less of the actual RGB value. To solve this problem we made both the pixel values equal meaning we lessen the RGB values till depth values. So this way we have come up with a functional representation,

$$DRPD = \left(\sum_{i=1}^{n} ((Xi-Ai) / Xi)*100 + \sum_{i=1}^{n} ((Yi-Bi) / Yi)*100 \right)/n \quad ....(1)$$

Here,

X, Y is the pixel values of depth picture and A, B is the RGB pixel values.

n is the total number of examples (for our thesis example case it was 5)

I is the sequence number.

DRPD will be a direct percentage value that contributes in decreasing the accuracy rate. For our case it was approximately 39%.

Software incompetence: We used matlab2016a software that is the updated version of matlab but there are others software's like visual studio that supports a lot of updated feature for plotting 3D graphical model. These updated technologies help rotating the model and view it more accurately. For matlab we could move our model flexibly, it could barely hold and was taking a very long time to execute one command. Moreover it doesn't support pictures with very large resolution.

Motion: While taking the input depth and RGB pictures an object must stay still during the process, otherwise Kinect sensor cannot correctly catch a moving object. In one of our example, that is Fig 3.9(a), the object who was one of our member, was moving so the model could not be as perfect as the other models.

Device placement: We used kinect sensor XBOX one. The two cameras (one is the RGB camera and the other is the depth camera) have a small portion of difference in between their centers. The accurate measurement of their difference cannot be measured as Microsoft did not state the accurate difference in between their centers. So fixing the object in one place and taking both the RGB and Depth pictures without moving the Kinect sensor, depth picture is sided than RGB. So while merging and ploting the two of them, it is displaced. So, these were the limitations that contribute in lessening the accuracy rate and for our case we measured most of it from the first factor DRPD which was 39%. So our accuracy rate should be close to 1-.39 = .61 which means 61%. If we could eliminate dependencies than accuracy rate should go high.

Conclusion

In this paper, we described a methodology to cost effectively plot 3D model of an object where both real and virtual objects case study was explained, though our research mainly focus on real objects. However, after achieving this there were still a few limitations, our proposed system can't make 3D in real-time, it takes few moment for creating 3D model. In our proposed system we used Microsoft Kinect sensor, it switches between 15 and 30 FPS depending on the lighting conditions. For that hardware limitation, our

system can not generate 3D model in real time. Besides, there are many scopes for improving our method. Our proposed system can create a 3D model taking images just from one angle. As this can be achieved only by an angel the images taken only once for depth and RGB without having to move the position of the sensor which will greatly help making 3D model with more accuracy. Also, we can see now a day's console games are expanding step by step and they mostly have these Kinect sensors for playing. This thesis will inspire them to utilize those Kinect to some use and join the era of demonstrating 3D model easily and do further research on 3D models which can lead them to a considerably brighter future. This thesis will also help in the field of animation and model designing as the methods of creating animation models now are extremely high costing. Moreover, by using our work we will try to develop it further by capturing and creating models from more angles makes and will also try to make the whole 3D scenario much easier and cheaper with less effort.

Acknowledgment

The authors wish to thank Md. Alium Basir and Fahima Akhter for their support in collecting 2D data.

References

[1] E.Lachat, H.Macher, M.A.Mittet, T.Landes, and P.Grussen Meyer, "First Experiences with Kinect v2 Sensor for Close Range 3D modelling," Technical Report Volume XL-5/W4, The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Feb. 2015.

[2] J.Vasky, M. Elias, P. Bezak, "3D Model Generation From the Engineering Drawing," *Research Papers Faculty of Materials Science and Technology Slovak University of Technology,* vol. 18, no. 29, pp. 47-53, available at https://content.sciendo.com/view/journals/rput/18/29/article-p47.xml, Mar. 2011.

[3] K.H. Choi, J.N. Hwang, "A real-time system for automatic creation of 3D face models from a video sequence", *Proc. 2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Apr. 2011, doi: 10.1109/ICASSP.2002.5745054

[4] N.Pears, Y.Liu, and P.Bunting, *Passive 3D Imaging.* 3D Imaging, Analysis and Applications, pp. 35-94, 2012. (Book style)

[5] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," *Proc. Third International Conference on 3-D Digital Imaging and Modeling*, Aug. 2002, doi: 10.1109/IM.2001.924423**.** (Conference proceedings)

[6] T. Yeh, J. Kim, "CraftML: 3D Modeling is Web Programming," *SIGCHI Human Factors in Computing (CHI 2018)*, pp. 1-12, Apr. 2018, doi: 10.1145/3173574.3174101**.** (Conference proceedings)

[7] Z. Zhu, C. Yan,L. Li, Y. Ren, Q. Luo, "Stereoscopic visualization of 3D model using OpenGL," *Proc. 2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP),* Nov. 2017, doi: 10.1109/GlobalSIP.2017.8309183 (Conference proceedings)