Construction of 3D Environment from 2D RGB Images and Depth Information

by

Redwanul Islam Shakir
19101052
Sarwar Hossain
19101187
Mehrab Mohsin Khan
19101378
Md Ishtiaq Enam Mayaz
19101112

A thesis submitted to the Department of Computer Science and Engineering in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science And Engineering

Department of Computer Science and Engineering
Brac University
May 2023

© 2023. Brac University
All rights reserved.

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Redwanul Islam Shakir
19101052

Sarwar Hossain
19101187

Mehrab Mohsin Khan
19101378

Md Ishtiaq Enam Mayaz
19101112

Approval

The thesis/project titled "Construction of 3D Environment from 2D RGB Images and Depth Information" submitted by

1. Redwanul Islam Shakir(19101052)
2. Sarwar Hossain(19101187)
3. Mehrab Mohsin Khan(19101378)
4. Md Ishtiaq Enam Mayaz(19101112)

Of Spring, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on May 30, 2023.

Examining Committee:

Supervisor:
(Member)

Mr. Rafeed Rahman
Lecturer
Department of Computer Science and Engineering
BRAC University

Co-Supervisor:
(Member)

Md. Ashraful Alam, PhD
Assistant Professor
Department of Computer Science and Engineering
BRAC University

Program Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD
Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
BRAC University

Abstract

Modern operations have introduced a new class of highly functional applications, redefining what can be built in a given amount of time. 3D modeling is required to resolve several shortcomings in outdated processes and enhance design team efficiency. Modern 3D modeling should be able to deliver design depth that 2D drawings or designs cannot. Furthermore, it should also allow engineers to experiment with physical components of a design without being constrained by physical constraints. This research paper proposes and demonstrates a 3D Construction method for creating a 3D Mesh by acquiring depth and color dataset from a real object using the Microsoft Kinect Sensor and HD Camera, which can then be extracted to applications such as game engines like Unity or other designing applications for the creation of the reference plane and, ultimately, the three-dimensional Environment Model.

Keywords: 3D-Model; 2D-Images; RGB; Depth; Microsoft Kinect; IR Camera; HD Camera; Construction; Sensors; Distance Vectors; three-dimensional Environment

Acknowledgement

Firstly, all praise to the Almighty Allah for whom our thesis have been completed without any major interruption.

Secondly, to our Supervisor Mr. Rafeed Rahman Sir and our Co-Supervisor Dr. Md. Ashraful Alam Sir for their kind support and advice in our work. They helped us whenever we needed help.

And finally to our parents without their throughout support it may not be possible. With their kind support and prayer we are now on the verge of our graduation.

Table of Contents

List of Figures

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

$2D$	Two-Dimensional
$3D$	Three-Dimensional
AE	Auto-Encoders
CAD	Computer- Aided Design
ICP	Iterative Closest Point
IR	Infrared
IRT	Infrared Thermography
NYU	New York University
RGB	Red, Green & Blue
$RGB-D$	Red, Green, Blue - Depth
SDK	Software Development Kit
$V3DOR$	Voxel-based 3D Object Reconstruction
VAE	Variational Auto-Encoders
VFX	Visual Effects

Chapter 1

Introduction

1.1 History of 3D Modelling

Starting from the 1950s, the computer-generated visuals were created by computer-aided design (CAD) software utilizing the limited 2D graphics technologies they had. The use of computerized 2D drawings and blueprints has expedited and enhanced the planning process. The idea and implementation of 3D computer-generated imagery dates back to the 1960s. The aforementioned graphics were employed within gaming contexts.

The advancement of 3D modeling software occurred during the 1970s. Individuals may generate progressively intricate three-dimensional models through the application of diverse techniques and approaches. The possibility of obtaining data from various sources was feasible due to the program's capabilities. At present, 3D printers enable individuals to transform their digital designs into tangible three-dimensional objects.

During the 1980s, the film and television industries were introduced to 3D modeling through "The Adventures of André and Wally B." This occurred concurrently with the release of the initial 3D-animated motion picture. Auto CAD and 3D Studio Max came out to help designers model the objects and environments.

In the next decade, there was a development of 3D modeling tools, like Blender, for personal computers and video games. These tools contributed to the change in 3D modeling architecture. Later, this widespread 3D modeling gave birth to virtual and augmented reality, which emerged during the 2000s.

3D modeling now has a massive number of applications that include animations, cinematography, architecture, gaming and more.

1.2 Problem Statement

Beginners usually struggle to create 3D mesh due to their technical complexity. This makes 3D construction difficult for beginners as it requires a lot of work and attention to detail and also being unfamiliar to the field, hence it takes a lot of time. Making 3D meshes rapidly is difficult for most designers. The Computer-Aided Design (CAD) software that is now available lacks the adequate capabilities to accurately map the quantitative measurements of objects found in the environment. Hence, the process of creating a 3D model involves a very large share of manual labor when linking specific points, approximately modeling curves and surfaces, and applying textures to surfaces[12]. 3D modeling demands expensive software and tools. This might be an impediment for beginners in 3D modeling.

1.3 Research Objectives

New approaches and algorithms are needed to build more accurate and efficient 3D models. Automating 3D model generation saves time and allows for detailed representations of massive items and surroundings. New ways for collecting, processing, and displaying enormous volumes of data, 3D models, and real-time rendering are needed to provide interactive experiences for Virtual and Augmented Reality, Gaming, Robotics, and other industries. Technology, information, and demand for three-dimensional meshes necessitate these approaches. Technology must develop for 3D engagement. Phone cameras can recreate real-world scenes. Faster 3D model creation, editing, and viewing.

We aim to make 3D model development more accurate, efficient, and automated, and we want to create novel methods to store and include massive quantities of data so our models may be utilised in more situations. These will help us create pragmatic and useful models.

1.4 Research Benefits

Numerous sectors, including gaming, robotics, architecture, and social media platforms, are placing a growing emphasis on diversified, elevated 3D content[14]. However, aesthetic modeling abilities and specialized technological expertise are required for the time-consuming manual generation of 3D products. With the use of 3D models, engineers and designers can spot and fix flaws before they lead to major blunders. For the benefit of architects, urban planners, and engineers, they may also model and reproduce real-world objects and settings. Video game characters, environments, and VFX may all benefit from 3D assets.

Moreover, in order to better comprehend and treat complex biological and healthcare systems[12], researchers may employ 3D assets to construct simulations and visualizations. Augmented and virtual reality applications in the fields of education, entertainment, and training rely on 3D models to generate convincing surroundings and interactions with the real world. Robots may be designed and tested in virtual

worlds using 3D models that imitate their movements and interactions. The use of 3D meshes in the design and planning stages of building projects has allowed architects and engineers to better envision the final result and spot possible problems before work even starts.

Therefore, 3D assets are frequently utilized to boost efficiency and lower costs in a number of industries for the reason that they enable greater perception, construction, and assessment ahead of the actual execution of a solution being carried out. This may be accomplished by making provision for enhanced analysis of a proposal prior to being successfully implemented in a system.

Chapter 2

Detailed Literature Review

In this section, we will explore various research papers pertaining to previous works done on 3D model creation, devices and techniques used, in order to obtain an understanding of our research field elaborately.

Kinect is equipped with a Red, Green, and Blue (RGB) Camera in addition to an Infrared (IR) Emitter and camera[3]. They are able to record colorful images as well as the depth of each pixel in the environment that is being seen. These data include information about the visual appearance of the frames as well as its geometry. They work well together and enable activities that would be challenging, if not inconceivable, if relied solely on visual representations. As a consequence of this, the data that is gathered by the Kinect (RGB & Depth) has an architecture that produces a new method of processing visuals that is referred to as RGB-D Image. A camera capable of capturing RGB light was used to create the colorful picture. The measuring of depth is carried out with the assistance of an infrared emitter and camera. However, the Kinect depth has various issues, such as noise and holes, which need some more complex filtering strategies.

This research[11] presents a novel neural network architecture that makes use of deep learning in order to do gait recognition using Kinect in a way that is both reliable and effective. The performance of the classifier is improved as a result of the training of the proposed neural network model using view- and pose-invariant feature vectors of dynamic joint relative cosine dissimilarity and joint relative triangular area. The iterative usage of the Adam optimization strategy may reduce the deterioration of the objective function. The suggested model is examined in contrast to a variety of different alternative models, some of which include the hyperbolic tangent activation function and the rectified linear unit activation function, amongst others. In particular, a comparison is made between the more conventional approaches to machine learning and the several gait detection systems that have been created in more recent times. The deep learning neural network that was recommended, when trained using the presented geometric attributes, attained the highest possible level of accuracy.

Furthermore, the introduction of Intel's RealSense D400 Series depth imaging technology marks a significant step forward since it provides a collection of low-cost, user-friendly 3D cameras that can be used in various environments[15]. In addition,

this cutting-edge technology is offered in an extensive range of adaptable hardware configurations, for understanding the convenience and the efficiency with which developers may create 3D vision applications using the Intel RealSense Software Development Kit (SDK). The application makes use of depth information in order to provide real-time data on the actual size of objects in a video stream that was captured by a RealSense depth cameras. The program used computer vision packages that are widely used, such as OpenCV, in conjunction with deep learning methods for the purposes of object localization and identification.

According to this article[6], the use of infrared thermography to determine how different items and surfaces are heated has proliferated across several sectors. If spatial information on the temperature distribution is provided using several infrared thermography (IRT) images collected by a 2D thermal imager to pinpoint the precise location of a heat source in 3D, the approach may be further extended to 3D applications. Thermal imaging collects data on temperature and other physical properties in three-dimensions. In order to accurately capture the shapes and colors of the environments under study, a color camera was used. The technique creates a 3D model from the thermal pictures using silhouette volume intersection. To evaluate the quality of the reconstructed 3D models, researchers contrasted the entropy technique and Otsu's approach, both of which rely on reprojection scoring. Reprojection rating is used to objectively compare the various methods' efficacy and precision.

This research[12] provides a temperature-based 3D reconstruction approach which uses multimodal blended images and Structure from Movement and Multi-view Stereo. Automated analysis of radiometric data to eliminate infrared interferences, boost false-color contrast, and multimodal co-registration under controlled conditions yielded infrared visuals. Quantitative investigations using IR and visible-light modalities aid sophisticated processing like 3D reconstruction, which is crucial in biological research. An efficient method of image fusion is able to extract information from the source data and include it in the combined images without introducing artifacts that might compromise the accuracy of the results. Excellent visible-light photographs and infrared images were required in order to reconstruct surfaces successfully in a significant and accurate manner. For the purpose of treating diabetic feet and making diagnoses, automatic processing of IR radiometric for 3D surface reconstruction of the foot was presented. This was shown using a temperature measurement. It eliminates the need for human processing and processes all data and picture arrays according to the same segmentation criteria, hence reducing the amount of time required and eliminating the possibility of user interpretation.

In the following paper[8], a quality Vector generative model of an object is utilized to both create a new 3D model and also enable perception from a 2D image. The study presented the TL-embedding network, which combines an autoencoder to guarantee a dynamic representation with a convolutional network to provide a predictable one.The extraction of 3D models and voxel prediction from 2D images are only two of the many jobs that are made possible as a direct consequence of this development. After applying convolution and deconvolution layers to encode the 3D voxel map to a low-dimensional domain, the autoencoder decodes a datapoint to a 3D mapping. The autoencoder ensures embedding so objects may be reproduced.

In order to train the TL-embedding network, it is required to have both 2D RGB photos and the 3D voxel maps that correspond to them. This is the case throughout the training phase of the network. Displaying CAD models against a diverse selection of random settings was used to generate voxel maps. This was done since there is such a little quantity of this data that is currently accessible.

The topic of this study[7] is the reconstruction of a 3D model from a single picture, and Point cloud coordinates, which is an easy-to-understand format, are how the final product is presented to the user. The developed architecture is a conditional shape sampler that is capable of predicting several realistic 3D point clouds from an input picture, despite the unconventional output form and the inherent discrepancy in shape for an input image. This is the case despite the fact that the output form is unconventional and the shape of an input image is inherently inconsistent. The point set generator architecture, which implements the point set prediction network, and the distance metric between point sets are the essential models that are employed in the generation of 3D models. Nevertheless, the most difficult aspects of working with this model are coming up with a way to represent unordered data using a three-dimensional point cloud and determining how best to handle certain ambiguities.

The Voxel Coloring Algorithm used in this study[2] necessitates a large amount of storage space for its calculation and voxel array, and it is not always successful in concealing an object's sharpness. Simpler and more powerful computers have contributed to the rise in popularity of volumetric reconstruction methods for 3D reconstruction. The consistency metric is crucial in Photo-Consistency based reconstruction. When we use threshold-based methods like original consistency check and adaptive consistency check, The quality of the image that is rebuilt is significantly impacted by the threshold value that we choose to utilize in the reconstruction process. There is still another method for evaluating voxel consistency that does not depend on a threshold value, and that method is the histogram consistency check.

In another similar work[13], the use of autoencoders (AE) and variational autoencoders (VAE) models for voxel-based 3D object reconstruction (V3DOR) from a single 2D picture is proposed for improved accuracy. From a single 2D picture, the encoder of both models learns an appropriate compressed latent representation, and the decoder creates the matching 3D model. There are three primary phases to the suggested technique. To begin, the encoder is fed the input 2D picture and uses it to learn the geometrical constraints in compressed form. Second, the latent representation of the input picture is acquired during encoding in the basic AE method. In contrast, the proposed 3D-VAEN method computes mean and standard deviation encoded vectors from input data during the encoding step. Third, a decoding procedure is executed to transform the acquired encoded information into a 3D model. For each of the two methods that have been presented, decoding works in the same way. The suggested method's quality is shown via the usage of IOU as a criterion of assessment. Unfortunately, satisfactory results from cross-data-set validation are not to be anticipated.

As proposed by this article[1],The affine deformation of the object that is seen in the observable picture under a weak perspective may be used to help quantify the degree to which a 2D image differs from a 3D model. This can be done by assigning a specific degree to the difference. Existing measurements that are derived in image space (image metrics), as well as metrics that are generated in transformation space (transformation metrics), can quantify the distance that separates the observed picture from the view of the item that is closest to it. Image metrics and transformation metrics both quantify this distance. The transformation metric that was used in the study may either be utilized to evaluate the image metric or directly examine the degree to which models and graphics are similar to one another. Both of these options are available to the researcher. In addition, the Alignment algorithm is a technique that examines the comparability of models and pictures based on the fewest feasible correspondences that exist between the two different kinds of media. This approach is known as the Minimum Correspondence Approach. Even though the use of few correspondences is advantageous for object recognition in polynomial time complexity while mitigating partial occlusion, it frequently leads to inaccuracies in the calculation of the distance between models and images.

Using laser scanning, geometric computer vision, and photogrammetry, this article[9] outlines generic methods for 3D data acquisition. It is possible to combine the created point clouds by making use of the most recent advancements in laser scanning, computer vision, photogrammetry, and statistical inference. For the purpose of capturing on-site 3D data on buildings and urban design, traditional terrestrial data collection and 3D modelling techniques, such as laser scanning and photogrammetry, are used. This allows for the creation of 3D building models that have fully textured surfaces. First, they need to be georeferenced with the assistance of control data, and then they need to be connected together. Through the use of vanishing lines and points, it is possible to manually reproduce the posture of the camera by using the interactive modelling approach, which is an extremely effective technique. This quest is made easier by the availability of numerous software packages for 3D modelling and processing units that are of higher quality.

Moreover, this study[14] suggests a novel approach to developing a high-quality generative model in three dimensions. As part of its computation, our model generates meshes and uses a fast (differentiable) graphics renderer. With the help of GET3D, regular 3D graphics engines can generate textured models. During training, the 3D model is rendered as 2D high-resolution images via a differentiable rasterizer. A 3D SDF and texture field were generated using two hidden codes. After that, networks of non-linear mapping nodes were developed. These templates control the creation of 3D shapes and textures. The model can produce shapes with any topology, high-quality textures, and many geometric elements, making the creation of 3D material accessible to a wider audience. A.I. However, in order to train, 2D silhouettes were used, and cameras were placed strategically. All of the evaluations of GET3D were conducted using simulated data. Models textured using GET3D are analyzed using pre-existing ShapeNet and Turbosquid datasets.

Chapter 3

Work Plan

To implement the proposed theory we decided to go through the following steps to get the desired data and adjust it according to our need to construct and refine a 3D mesh before implementing it in a game engine or CAD.

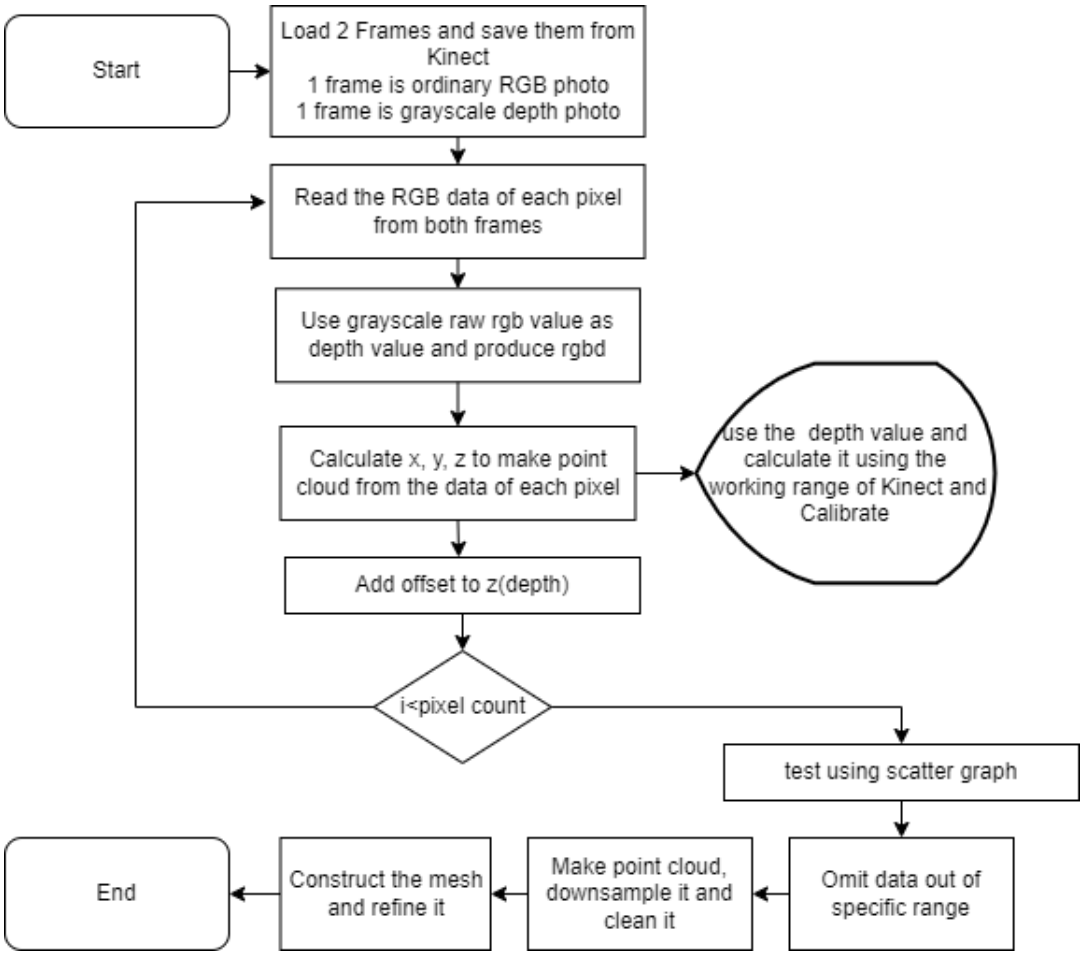

Figure 3.1: Flowchart Diagram of Work Plan

Chapter 4

Model Implementation & Analysis

4.1 Description of the Model

We used Processing 4 to frame capture 2 images from Kinect 1414 of XBox360 using the library kin4win32. The Kinect of Microsoft is a discontinued device[3], therefore we had to use 3rd party softwares and libraries to connect and get the data from the Kinect. The two frames (RGB & Grayscale depth image) we used out of many of our test models taken from NYU-Depth V2[4] dataset are shown below:

Figure 4.1: RGB Image

Figure 4.2: Grayscale Depth Image

The images are slightly out of sync. The first task is to get them aligned. We can use face detection to align the images[5] or manually crop them to align.

We extracted the raw RGB values of both images. The image is a grayscale image and it provides all the RGB values of 1 pixel as the same value. The value is between 0 to 255, farthest to closest, is also called depth value. The minimum range of the kinect is 80cm and maximum is 400cm[3]. The range is 320cm, the difference between maximum and minimum. The maxdepth is the maximum range of the kinect sensor. The following formula shows the depth calculation:

$$z = maxdepth - \frac{range * depthvalue}{256} \tag{4.1}$$

The i, j are the position of the pixel in 2D, focal is the actual focal length of the depth sensor, xResolution is the resolution of the image on x-axis and yResolution is the resolution of the image on y-axis. The following formulae are required for the calculation:

$$x = \frac{i - \frac{xResolution}{2}}{focal_x} \tag{4.2}$$

$$y = \frac{j - \frac{yResolution}{2}}{focal_y} \tag{4.3}$$

Thus, the x, y, z of the 3D environment is calculated via a very simple equation. This algorithm has a running time of O(n) where it runs the number of times that equals the total number of pixels.

4.2 Description of the Data

The data we managed to pull out consists of x,y,z coordinates of each point to be drawn in unity along with the RGB raw values and raw depth value providing a raw RGB-D dataset with processed x, y, z coordinates. If the image is 640x480, the dataset will contain 640 times 480 data, which is 3,07,200 pixels with their color value and point coordinates. The data looks like the following:

	A	B	C	D	E	F
1	red	green	blue	x	y	z
2	253	254	255	-26.666666666666668	-35.55555555555556	-65.375
3	248	249	253	-26.666666666666668	-35.44444444444444	-65.375
4	253	254	255	-26.666666666666668	-35.333333333333336	-64.88888888888889
5	253	254	255	-26.666666666666668	-35.22222222222222	-65.0625
6	244	246	247	-26.666666666666668	-35.111111111111114	-64.57142857142857
7	247	249	250	-26.666666666666668	-35.0	-64.57142857142857
8	253	255	255	-26.666666666666668	-34.888888888888886	-65.375
9	249	251	252	-26.666666666666668	-34.77777777777778	-64.92857142857143
10	252	254	254	-26.666666666666668	-34.666666666666664	-64.125
11	253	255	255	-26.666666666666668	-34.55555555555556	-64.59375
12	253	255	255	-26.666666666666668	-34.44444444444444	-64.59375
13	253	255	255	-26.666666666666668	-34.333333333333336	-65.375
14	253	255	255	-26.666666666666668	-34.22222222222222	-64.75
15	253	255	255	-26.666666666666668	-34.111111111111114	-64.88888888888889
16	251	255	255	-26.666666666666668	-34.0	-64.125
17	250	255	253	-26.666666666666668	-33.888888888888886	-64.4375
18	255	255	254	-26.666666666666668	-33.77777777777778	-64.125
19	255	255	252	-26.666666666666668	-33.666666666666664	-65.86111111111111
20	255	255	252	-26.666666666666668	-33.55555555555556	-66.3125
21	255	255	252	-26.666666666666668	-33.44444444444444	-66.20833333333333
22	255	255	252	-26.666666666666668	-33.333333333333336	-66.17857142857143
23	255	255	252	-26.666666666666668	-33.22222222222222	-66.25
24	255	255	252	-26.666666666666668	-33.111111111111114	-66.0
25	255	255	252	-26.666666666666668	-33.0	-65.84375
26	255	255	252	-26.666666666666668	-32.888888888888886	-65.6875
27	255	255	252	-26.666666666666668	-32.77777777777778	-67.25
28	255	255	252	-26.666666666666668	-32.666666666666664	-66.41666666666667
29	255	255	252	-26.666666666666668	-32.55555555555556	-65.82142857142857
30	255	255	252	-26.666666666666668	-32.44444444444444	-65.375
31	255	255	252	-26.666666666666668	-32.333333333333336	-66.17857142857143
32	255	255	252	-26.666666666666668	-32.22222222222222	-66.15625
33	255	255	252	-26.666666666666668	-32.111111111111114	-65.6875

Figure 4.3: Raw Data from Dataset

4.3 Preliminary Analysis

After obtaining the coordinate values, we did an analysis to test the data by scattering the data on a 3D plane using matplotlib. The 3D visualization was what we were expecting. The following diagrams demonstrate the scatter plot of the data.

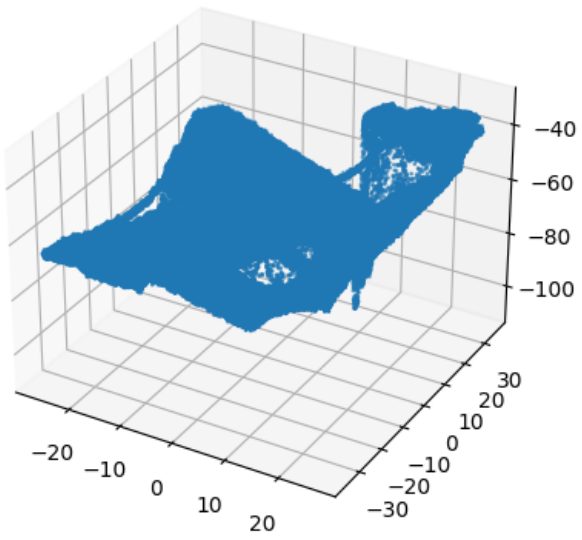

Figure 4.4: Scatter Plot of Data (Top View)

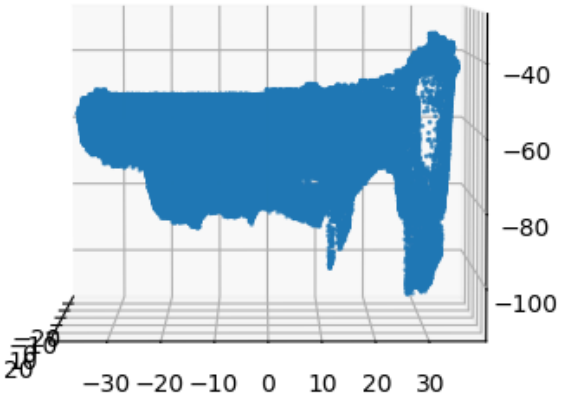

Figure 4.5: Scatter Plot of Data (Side View)

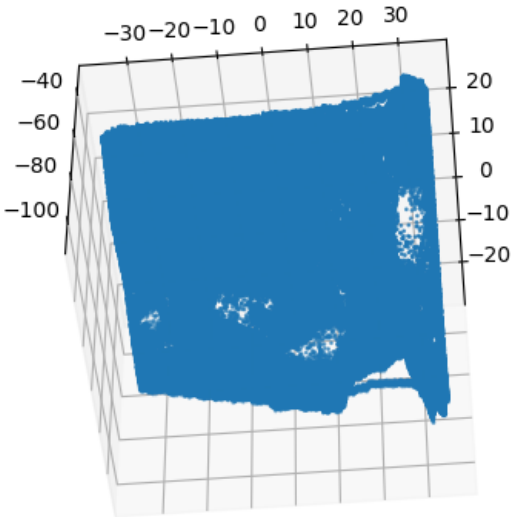

Figure 4.6: Scatter Plot of Data (Front View)

Chapter 5

Construction of the Mesh

5.1 Point Cloud

All the RGB values are taken into an array and the x,y,z, are taken into another array which is called points. We use Open3D[10] to draw the geometry, make the mesh and also do some post processing. This is a library that helps to develop programs that deal with 3D. Open3D contains sets of python and C++ algorithms that help process the point cloud and to visualize them.

A point cloud was created using Open3D geometry. The x, y, z points and RGB color values are added to the point cloud. The point cloud can be visualized using Open3D's visualization and one example is shown in Figure 5.1:

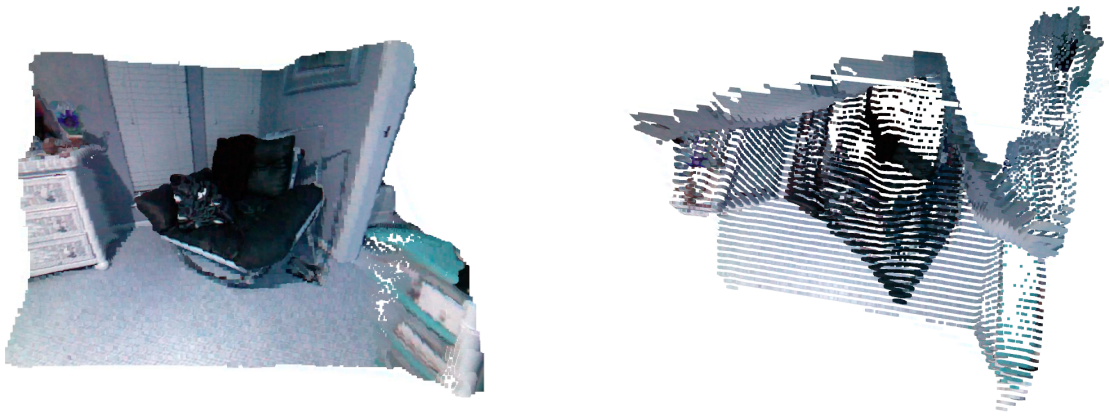

Figure 5.1: Front and Top view of Point Cloud Visualization

5.2 Introducing offset to depth

It can be seen in Figure 5.1 that the front view is quite detailed while the top view looks like a set of lines. This is because of the low accuracy of the sensor. This will impact negatively while creating a mesh. To overcome this, an offset is introduced to the depth which will smoothen the depth. The value is between 0 to 255, farthest to closest, is also called depth value. The minimum range of the kinect is 80cm and maximum is 400cm[3]. The range is 320cm, the difference between maximum and minimum. The randint(10) takes a random integer from 0 to 10. Offset is calculated by the following formula:

$$offset = \frac{range * depthvalue}{256 * randint(10)} \tag{5.1}$$

$$z = z \pm offset \tag{5.2}$$

This offset displaces some points by a random small proportion of depth per bit on the z-axis. This causes the linings in the point cloud to smoothen. Here is the visualization after introducing offset:

Figure 5.2: Front and Top view of Point Cloud Visualization after adding offset

5.3 Filtering and cleaning the points

Filtering and cleaning is done by Open3D. The neighbors of the points are cleaned by removing statistical outliers. The cloud is downsampled using voxel downsampling. Then radius outliers of the point neighbors are filtered. The normals of the downsampled cloud is estimated using KDTreeSearchParamHybrid of Open3D, and afterwards the normals are oriented along the tangent of the plane. A filtered cloud point is obtained that is ready to be constructed to a mesh.

5.4 Mesh Construction and Refinement

Surface Reconstruction from Open3D is used on the filtered cloud point with normals to make a triangle mesh. A few post processing is done on the mesh to clean it. Laplacian smoothing and taubin smoothing is done on the mesh. Duplicate and degenerate triangles are removed, duplicate vertices are removed and non manifold edges are removed. Finally, after all the refinements are done, the mesh is ready to be displayed and saved to be used later for different purposes. The visualization of Open3D is now used to visualize the mesh:

Figure 5.3: Front and Side View of Mesh visualization

5.5 Mesh Comparison with and without offset

The comparison is shown below in Fig 9. The mesh without offset has more lines like structure present at specific displacements in the depth (z) axis, making it less like the original object. Meanwhile the mesh with offset added to depth shows a significant improvement and the mesh looks more refined although including this offset causes it to lose a few details.

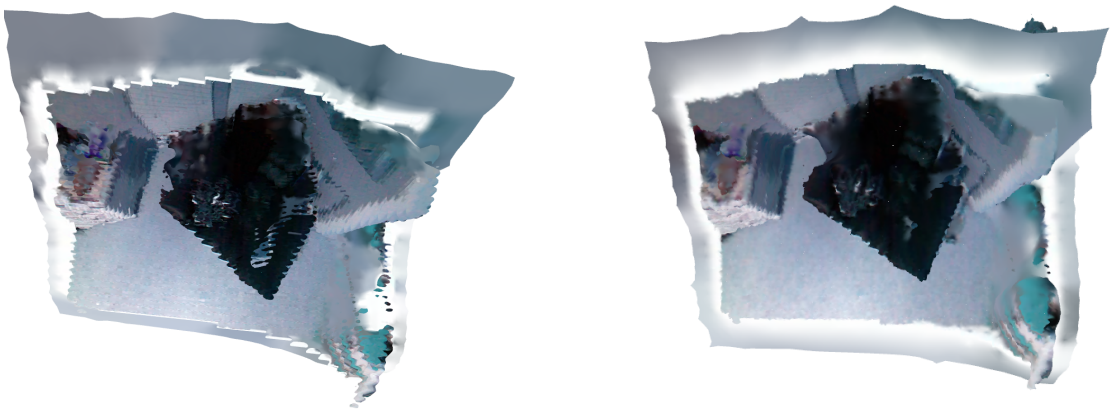

Figure 5.4: Top view of mesh with offset(right) and without offset(left)

Chapter 6

Conclusion

After the collection of data, pre-processing, analysis and mesh creation, we can see that the proposed theory of producing a 3D environment from just a RGB image and its corresponding Grayscale Depth image is feasible. RGB image can be taken by any ordinary camera while the depth image can be taken by a Kinect or Lidar or IR camera. Our depth calculation approach might be used to determine the depth of each pixel in order to produce a three-dimensional representation. This method makes the process of making standard 3D objects and environments considerably faster and simpler. More improvements can be done to the mesh by using a depth and RGB sensor of higher resolution and better focal length. Taking more photos from various angles and applying registration on all the meshes using ICP(Iterative Closest Point) may result in a finer and more accurate mesh.

The sensors needed are available to most of the smart mobile phones around, like phones these days usually have cameras more than 40 megapixel and some phones come with Lidar sensor, one of the best depth sensors. They will be able to replicate an environment just that easily with very high render quality.

To conclude, it can be said that this proposed theory will be of great help to 3D artists, game developers, AR and VR developers and architects.

Bibliography

[1] R. Basri and D. Weinshall, "Distance metric between 3d models and 2d images for recognition and classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 4, pp. 465–479, 1996. DOI: 10.1109/34.491630.

[2] K. Susheelkumar, V. Semwal, S. Prasad, and R. Tripathi, *Generating 3d model using 2d images of an object*, Jan. 2011.

[3] L. Cruz, D. Lúcio, and L. Velho, "Kinect and rgbd images: Challenges and applications," *2012 25th SIBGRAPI Conference on Graphics, Patterns and Images Tutorials*, pp. 36–49, 2012.

[4] P. K. Nathan Silberman Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012.

[5] R. Min, N. Köse, and J.-L. Dugelay, "Kinectfacedb: A kinect database for face recognition," *Systems, Man, and Cybernetics: Systems, IEEE Transactions on*, vol. 44, pp. 1534–1548, Nov. 2014. DOI: 10.1109/TSMC.2014.2331215.

[6] C.-Y. Chen, C.-H. Yeh, B. Chang, and J.-M. Pan, "3d reconstruction from ir thermal images and reprojective evaluations," *Mathematical Problems in Engineering*, vol. 2015, pp. 1–8, Aug. 2015. DOI: 10.1155/2015/520534.

[7] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3d object reconstruction from a single image," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2463–2471, 2016.

[8] R. Girdhar, D. F. Fouhey, M. D. Rodriguez, and A. K. Gupta, "Learning a predictable and generative vector representation for objects," *ArXiv*, vol. abs/1603.08637, 2016.

[9] D. Fritsch and M. Klein, "3d and 4d modeling for ar and vr app developments," in *2017 23rd International Conference on Virtual System Multimedia (VSMM)*, 2017, pp. 1–8. DOI: 10.1109/VSMM.2017.8346270.

[10] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.

[11] A. Bari and M. Gavrilova, "Artificial neural network based gait recognition using kinect sensor," *IEEE Access*, vol. PP, pp. 1–1, Nov. 2019. DOI: 10.1109/ACCESS.2019.2952065.

[12] R. Bayareh Mancilla, B. Tấn, C. Daul, *et al.*, "Anatomical 3d modeling using ir sensors and radiometric processing based on structure from motion: Towards a tool for the diabetic foot diagnosis," *Sensors*, vol. 21, p. 3918, Jun. 2021. DOI: 10.3390/s21113918.

[13] R. Tahir, A. B. Sargana, and Z. Habib, "Voxel-based 3d object reconstruction from single 2d image using variational autoencoders," *Mathematics*, vol. 9, p. 2288, Sep. 2021. DOI: 10.3390/math9182288.

[14] J. Gao, T. Shen, Z. Wang, *et al.*, *Get3d: A generative model of high quality 3d textured shapes learned from images*, Sep. 2022. DOI: 10.48550/arXiv.2209.11163.

[15] Hickeys, *Kinect for windows - windows apps*. [Online]. Available: https://learn.microsoft.com/en-us/windows/apps/design/devices/kinect-for-windows.

Code Snippet

Conversion to RGB-D Data

```
RGB = cv2.imread('RGBImage.jpg')
depth = cv2.imread('DepthImage.png')
x=[]
y=[]
z=[]
RGBD=[len(depth)][]
for i in range(len(depth)):

    for j in range(len(depth[i])):

        depthbit=depth[i][j][0]
        z_Calc=depthCalculation(depthbit)
        x_Calc=(i-len(depth)/2)/focal
        y_Calc=(j-len(depth[i])/2)/focal
        x.append(x_Calc)
        y.append(y_Calc)
        z.append(z_Calc)
        data=[RGB[i][j][0], RGB[i][j][1], RGB[i][j[2],
        x_Calc, y_Calc, z_Calc]
        RGBD[i].append(data)
```

Depth Calculation

```
def depthCalculation(depthbit):
    workingRange=maxRange-minRange
    depthPerBit=workingRange/256
    z=maxRange-(depthbit*depthPerBit)+offset(depthPerBit)
    return z
```

Offset Calculation

```
def offset(ofRange):
    rand=random.randrange(-10,10)
    if rand==0: return rand
    ofs=ofRange/rand
    return ofs
```