3D Mesh Reconstruction from Point Clouds

ILTIMAS KABIR
DARRYLL FONSECA
AISHIK SEN
KARAN CHAVAN

Period of work:

31.05.24 – 12.07.24

—

Under the guidance of

Dr. R S Sengar

Division of Remote Handling and Robotics (DRHR)

Bhabha Atomic Research Centre, Mumbai

DECLARATION

We hereby declare that the work described in this dissertation titled **"3D MESH RECONSTRUCTION FROM POINT CLOUDS"** forms our own contribution to the research work carried out under the guidance of **Dr. Ratnesh Singh Sengar**, Scientific Officer(H), Department of Remote Handling and Robotics (DRHR), Bhabha Atomic Research Centre (B.A.R.C)

Signature of the candidates:
ILTIMAS KABIR

DARRYLL FONSECA

AISHIK SEN

KARAN CHAVAN

In my capacity as the Principal Investigator/Supervisor of the candidate's dissertation, we certify that the above statements are true to the best of our knowledge.

Signature

Shri Dr. Ratnesh Singh Sengar
Scientific Officer (H)
Division of Remote Handling and Robotics (DRHR)
Bhabha Atomic Research Centre, Mumbai

ACKNOWLEDGEMENT

With immense pleasure and deep sense of gratitude, we wish to express our sincere thanks to our project mentor **Dr. Ratnesh Sengar**, Scientific Officer(H), Department of Remote Handling and Robotics (DRHR), Bhabha Atomic Research Centre (B.A.R.C), Mumbai, for giving us the project and guiding us through the whole process over the course of it. We would also like to extend our sincere thanks to **Mr. Sourav Mishra** and **Mr. Rohit Sharma**, Scientific Officer (E), Department of Remote Handling and Robotics (DRHR), Bhabha Atomic Research Centre (B.A.R.C.), without their motivation and continuous encouragement, this project work would not have been successfully completed.

Place: Mumbai
Date: 12/07/2024

ABSTRACT

The project focuses on reconstructing 3D meshes from 2D point clouds, which are derived from pre-existing 2D RGB images and their corresponding depth maps. This work employs various computational methods and surface reconstruction algorithms, including Poisson Surface Reconstruction, Ball-Pivoting Algorithm (BPA), and Delaunay Triangulation. Through comprehensive experimentation and analysis, we determined that Poisson Surface Reconstruction is the most effective method, followed by Delaunay Triangulation. These algorithms successfully produced watertight surfaces and effectively filtered out noise from the original dataset. The potential applications of our findings are diverse, encompassing the reconstruction of historical artifacts and monuments, architectural visualization, crack detection in walls or pipes, and object detection for robotic navigation. Our study aims to provide in-depth insights into the utility of converting 2D images to 3D representations for various applications in computer vision and 3D modelling. The findings highlight the significance of surface reconstruction algorithms in enhancing the accuracy and reliability of 3D models derived from 2D data, thereby advancing the field of computational modelling and its practical applications.

Keywords: Point Cloud, Poisson Surface Reconstruction, Ball-Pivoting Algorithm, Delaunay Triangulation, depth maps.

CONTENTS

1. INTRODUCTION

In the field of computer vision and surface reconstruction, the production of a 3D mesh of a surface from a 2D image is a pivotal problem, having applications in architectural reconstruction, detection of defects in walls or pipes and reconstruction of historical artefacts and monuments. Our project explores a detailed workflow to convert 2D images into a 3D mesh outlining their surface, using sophisticated algorithms such as Poisson Surface Reconstruction, Ball Pivoting Algorithm and Delaunay Triangulation.

We made use of the NYU depth V2 dataset that had .mat files containing RGB images and its corresponding depth map. The depth maps were converted to point clouds (.png) from which we generated polygon file format (.ply) containing the mathematical information about the surface on a pixel-by-pixel basis. A point cloud is a discrete set of data points in space which may represent a 3D shape, surface, or object. To convert these images into 3D meshes of the surface we used three distinct algorithms, each producing its own reconstruction of the object's surface:

- **Poisson Surface Reconstruction:** This technique offers a robust method of producing smooth and watertight surfaces from noisy and incomplete point clouds. This is very advantageous when making a mesh out of real-world objects where it is natural to have flawed point clouds due to noise, interference, and incomplete sampling. As per[1] the algorithm constructs an indicator function for the surface and calculates the gradient field from the point normal at each point. The Poisson equation can be written as:

  $$\boldsymbol{\nabla}\emptyset = \boldsymbol{\nabla}.\boldsymbol{V}$$

  Where $\boldsymbol{\nabla}$ is the Laplacian operator, $\emptyset$ is the scalar field and $\mathbf{V}$ is the vector field derived from the normal vectors.

  Upon solving the Poisson equation for the gradient field, we get a continuous scalar field whose gradient approximates the normal vectors. The resultant iso-surface provides a detailed and accurate 3D representation of the object in the original 2D RGB image. This method also smooths out noise and irregularities in the point cloud, resulting in a clean and continuous surface.

- **Ball Pivoting Algorithm:** This algorithm[2] constructs a surface by rolling a virtual ball over the point cloud and forming triangles with points on the ball's surface. This method is suitable for creating meshes from dense and uniformly distributed point clouds. BPA is based on the concept of rolling a ball of a fixed radius $\boldsymbol{r}$ over the point cloud and using the ball's traversal to find triplets of points to make more triangles. Starting from an initial seed triangle, the ball pivots around an edge of the triangle to find a new point such that the ball touches three points (forming a new triangle) and no other points lie inside the ball. The process continues until no new triangles can be formed. BPA emphasises local surface topology and

[1] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson Surface Reconstruction," in Eurographics Symposium on Geometry Processing, K. Polthier and A. Sheffer, Eds., 2006.
[2] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, "The Ball-Pivoting Algorithm for Surface Reconstruction," IEEE Trans. Vis. Comput. Graph., vol. 5, no. 4, pp. 349-359, Oct.-Dec. 1999.

connectivity, ensuring that the generated surface is as accurate as possible with respect to the original image.

- **Delaunay Triangulation:** This process creates triangles on the surface with the condition that the circumcircle of each triangle contains no points within it. Delaunay Triangulation tries to maximize the minimum angle of the triangles in the mesh which makes sure that triangulation of the surface is proper. The algorithm starts with the point cloud and constructs triangles while making sure that the Delaunay condition is met for each constructed triangle. First the algorithm makes a super triangle that encompasses all the points and join the vertices of the triangle to the point to form triangles. The circumcircle of each triangle is drawn and checked for the Delaunay condition. If the condition is violated the common edge is deleted and smaller triangles are constructed by joining the point violating the condition. The process is repeated until all triangles follow the Delaunay condition. The super triangle is deleted along with all of the edges connected to it, which gives us the resultant mesh of the point cloud. This ensures that the generated 3D mesh is uniform and has no holes. Delaunay Triangulation refines the produced mesh and enhances its structural integrity and quality.

For each mesh we also created a corresponding heatmap to display the depth (distance from the camera) of each item in the mesh. This was done to give more clarity to each surface and to clearly display that the mesh generated is an accurate 3D representation of a 2D image of the subject matter.

Each method has its own strengths and is chosen based on the specific requirements of the task such as the density and the distribution of the point cloud. Our paper provides a comprehensive comparison and analysis of these methods and illustrates their application on point cloud data derived from 2D RGB images. The primary contribution of the project is twofold, first the integration and evaluation of multiple 3D reconstruction algorithms on point cloud data generated from 2D images and second, an analysis of the resulting 3D meshes in terms of accuracy, efficiency, and surface quality. Our findings demonstrate the pros and cons of each approach, providing valuable insight for future research and practical applications in 3D surface reconstruction.

In the following sections we delve into the methodology of generating point clouds from 2D RGB images, using the point cloud data to make polygon format files containing all the data about the surface. We then describe each surface reconstruction algorithm in detail, and present a comparative analysis of the results obtained. Through this project we aim to understand, compare, and broaden the capabilities of 3D surface reconstruction algorithms.

2. LITERATURE REVIEW

The papers we referred had variety of methods to perform surface reconstruction from point clouds, each addressing the fundamental challenges in computer vision and computational geometry.

Nguyen et. al. in his paper [16] provides an overview of various techniques for surface reconstruction from point clouds. It categorizes methods into explicit and implicit approaches. Their insight into artifacts leads to the realization of several key properties of the prospective system: (1) the model should make predictions based on not only local features but also high-level semantics, (2) the model should consider spatial correlation between points, and (3) the method should be scalable, i.e., the output point cloud can be of arbitrary size. To inherit all these properties, they propose to approach the problem in two steps: feature blending and deformation. They dub the proposed method Point Cloud Deformation NETwork (PCDNet) for brevity.

Juszczyk et. al. in their paper [4], collected data from 29 measurement sessions with 21 patients with the patients age ranging from 34 to 82. Through the collected dataset, it created point clouds and from those point cloud it generated the surface reconstruction using the Poisson Surface Reconstruction. Region of interest (ROI) required for further analysis is defined as the healthy skin around the wound with the wound area in the central part. Their approach is dedicated mainly to wounds located on the upper and lower limbs; however, they believe it can be useful in other localization. The only limitation of this method is that the wound area must be in the field of view of the imaging devices.

Berger et. al in his paper [6] presents surface reconstruction algorithms from the perspective of priors: assumptions made by algorithms to combat imperfections in the point cloud and to eventually focus what information about the shape is reconstructed. Without prior assumptions, the reconstruction problem is ill-posed; an infinite number of surfaces can pass through (or near) a given set of data points. Assumptions are usually imposed on the point cloud itself, such as sampling density, level of noise and misalignment.

Gopi et. al in his paper [13] assumed that the inputs to the surface reconstruction algorithm are sampled from an actual surface (or groups of surfaces). If the surface is improperly sampled, the reconstruction algorithm can produce artifacts. They used 2D local Delaunay triangulation around a specific point to find the neighbors of that point.

Brüel-Gabrielsson et. al in his paper [5] wanted to characterize the shapes by several markers that are representational and explain their different properties. They believed that the algorithm could be used in many applications like shape alignment, shape correspondence, surface reconstruction and shape synthesis. They used Poisson surface reconstruction in their project. They build the surface to be higher-valued as opposed to lower-valued.

Abdulqawi et. al in his paper [17] followed the rules of reverse engineering. This comprises of taking existing models into considerations and modifying those models to create a new product. In industries where the 3D scanner scans the 3D object and creates the 2D images for that object. They decided to reverse this process for which they decided to use CAD/CMM model. The generated point cloud is saved in the format of .xyz file so that the 3D CAD software will be able to read and process the 3D point cloud.

Zhao et. al in their paper [9] proposed a new method to construct the polygon mesh from point cloud dataset based on the ball-pivoting algorithm. The core idea of this project was to determine a suitable radius of the pivoting ball according to the principles of the density self-adaption and balance distribution, which will greatly improve the accuracy and efficiency of the polygon mesh reconstruction from the point cloud data, especially from uneven point cloud data.

Bernardini et. al in his paper [7] worked on the simple principle of the BPA (Ball Pivoting Algorithm) that is the three points will form a triangle if the ball of a user-specified radius ρ touches them without containing any other point inside the circle. The process continues until the circle rolls through all the edges of the triangles. The process can then be repeated with circle of larger radius to handle uneven sampling densities. Regardless of the defects in the data, the BPA is guaranteed to build an orientable manifold.

Cazals et. al in their paper [12] aimed to construct a triangulated surface that can be directly used by downstream computer programs for further processing. Reconstruction step consists of two steps. First a piece-wise linear surface is reconstructed, and second, a piecewise-smooth surface is built upon the mesh. They used Delaunay triangulation for this reconstruction process. It seems that the Delaunay triangulation explores the neighborhood of a sample point in all relevant directions in a way that even accommodates non-uniform samples.

Guo et. al in their paper [15] created the kd-tree index for the point cloud data. Then, the Delaunay Triangulation of multicore parallel computing was used to construct those point clouds to the leaf nodes. Finally, the complete 3D mesh model was constructed by connecting the points. A mathematical method was used to divide the 3D space into tetrahedra (4-faced 3D shapes) while respecting certain constraints. It breaks the overall surface to smaller, manageable linear pieces after which the pieces are placed together, ensuring the best possible connections between the points.

Jiang et. al in his paper [10] approached towards their project by choosing α-shapes based and ball pivoting algorithm-based surface reconstruction techniques. α-shapes has an accurate shape approximated the original resampling point cloud, but exists large unwanted components. The ball pivoting algorithm was selected to reconstruct the topological graph, as the principle is simple, the ball with given radius can pivot along an existing edge to touch another point to shape a new triangle. Finally, the garment surface that was constructed by the ball pivoting algorithm has regular and uniform visual effects.

Rakotosaona et. al in his paper [18] combined the advantages of the classical methods with learning-based data priors. Their method is based on blending together Delaunay surface elements, which are defined by a 2D Delaunay triangulation of a local neighborhood in the point set after projecting to a planar 2D domain. They proposed an approach that predicts a local projection via learned logarithmic maps and use them to propose likely triangles using local Delaunay triangulation.

3. METHODOLOGY

3.1 Data Preparation and Preprocessing

The first step in our workflow involved the acquisition and preparation of data from the NYU Depth Dataset V2. This dataset is a comprehensive resource that had .mat files with paired RGB images and corresponding depth maps, capturing indoor scenes. The depth maps provide the necessary depth information, indicating the distance of each pixel from the camera, which is crucial for 3D reconstruction. First, we extracted the RGB images and depth maps from the .mat files. The RGB images are standard colour images, while the depth maps are textured images where the intensity of each pixel corresponds to its depth value. These extracted images serve as the foundational data for the subsequent steps of our methodology.

Once the RGB images and depth were extracted, we converted the depth maps into point clouds, in polygon file format (.ply) files. In our case, each point in the cloud corresponded to a pixel in the 2D depth map, with the depth value indicating its distance from the camera. To generate the point cloud, we mapped each pixel in the depth map to a 3D coordinate. The process involved converting the 2D pixel coordinates (x, y) and the depth value (z) into 3D space. The depth value was directly taken from the intensity of the pixel in the depth map, while the x and y coordinates were derived from the pixel's position in the image.

After generating the point clouds, we store them in two formats: .png and. ply. The .png format was used for visualization purposes, allowing us to visually inspect the point clouds and ensure their correctness. The. ply (Polygon File Format) were used in the surface reconstruction phase to create the 3D meshes. Below are the three main algorithm we used for generating the 3D meshes in detail.

3.2 Poisson Surface Reconstruction

Poisson Surface Reconstruction is a technique used to reconstruct a surface from a point cloud by solving a regularized optimization problem called a Poisson equation, to obtain a smooth surface. In our paper, we have used Poisson Surface Reconstruction to generate the 3-D meshes of the depth images. The input was polygon file format (.ply) files, as the primary input for Poisson Surface Reconstruction are point clouds. We can show that surface reconstruction from oriented points can be cast as a spatial Poisson problem. The methodological details of the process are given below including the mathematical equations.

1. Input Data

Here in our paper, we start with a set of points $\boldsymbol{P}$ from a point cloud having a set of sample points $\boldsymbol{p_i} \in \boldsymbol{P}$. Each point ($\boldsymbol{p_i}$) is represented by its co-ordinates ($\boldsymbol{x_i}, \boldsymbol{y_i}, \boldsymbol{z_i}$). For each point ($\boldsymbol{p_i}$), the inward-facing normal vector ($\boldsymbol{n_i}$) is calculated, which could be done by using Principal Component Analysis (PCA). The normal vector ($\boldsymbol{n_i}$) is the Eigenvector corresponding to the smallest Eigenvalue of the covariance matrix of the neighbours. Our goal here is to reconstruct a watertight, triangulated approximation to the surface by approximating the indicator function of the model and extracting the isosurface. The key challenge is to accurately compute the indicator function from the samples. In this section, we derive a relationship between the gradient of the indicator function and an integral of the surface normal field. We then approximate this surface integral by a summation over the given oriented point samples. Finally, we reconstruct the indicator function from this gradient field as a Poisson problem. The next step would be to generate a Vector Field (*V*) from the point cloud and the normal.

2. Vector Field Construction

A continuous Vector Field (*V*) from the point cloud and normal is constructed, which is defined over the entire domain of the points. Since the indicator function is a piecewise constant function, explicit computation of its gradient field would result in a vector field with unbounded values at the surface boundary. To avoid this, we convolve the indicator function with a smoothing filter and consider the gradient field of the smoothed function. The vector field V is such that it is close to the normal vectors at the points:

$$V(p_i) \approx n_i$$

Our goal here is to approximate a continuous field that reflects the sampled surface's gradient. For a point p_i with normal vector n_i,

$$V(p_i) = n_i\, \delta(p - p_i)$$

Here δ is the Dirac Delta Function which is used to create a vector field that accurately represents the sampled surface's gradient at the locations of the points in the point cloud. It is a mathematical construct that is zero everywhere except at p_i, where it is infinitely large, in such a way that its integral over the entire space is 1. Using the Dirac delta function, the vector field V can be expressed as:

$$V(p) = \sum_i n_i\, \delta(p - p_i)$$

3. Creating the Poisson Equation

Having formed the Vector Field ($\vec{V}$, we now solve for the function χ such that $\nabla\chi = \vec{V}$.

χ is the scalar function (implicit function) whose gradient best approximates a vector field $\vec{V}$ defined by the samples, i.e. $\min_\chi \| \nabla\chi - \vec{V} \|$. After applying the divergence operator, this variational problem transforms into a standard Poisson problem: computing the scalar function χ whose Laplacian (divergence of gradient) equals the divergence of the vector field $\vec{V}$,

$$\Delta\chi \equiv \nabla\cdot\nabla\chi = \nabla\cdot\vec{V}$$

However, in practice, $\vec{V}$ is constructed such that

$$\nabla\cdot\vec{V} \approx \sum_i n_i\, \delta(p - p_i)$$

Formulating surface reconstruction as a Poisson problem offers several advantages. Many implicit surface fitting methods segment the data into regions for local fitting, and further combine these local approximations using blending functions. However, $\vec{V}$ is generally not integrable (i.e. it is not curl free), so an exact solution does not generally exist. To find the best least-squares approximate solution, we apply the divergence operator to form the Poisson equation

$$\Delta\chi = \nabla\cdot\vec{V}$$

In the next section, we have described the process of solving the Poisson Equation in detail.

4. Solving the Poisson Equation

The Poisson equation we need to solve is $\Delta\chi = \nabla\cdot\vec{V}$**.** To solve the equation numerically, we need to discretize the domain into a regular grid. Now, considering a 3D grid where each grid point is indexed by *(i,j,k),* we can create the Discretized Laplace equation, and $\Delta\chi$ can be approximated at the points using finite differences as:

$$(\Delta\chi)_{i,j,k} \approx \frac{(\chi)i+1,j,k-(2\chi)i,j,k+(\chi)i-1,j,k}{\Delta x^2} + \frac{(\chi)i,j+1,k-(2\chi)i,j,k+(\chi)i,j-1,k}{\Delta y^2} + \frac{(\chi)i,j,k+1-(2\chi)i,j,k+(\chi)i,j,k-1}{\Delta z^2}$$

For simplicity, assuming uniform grid spacing, i.e: $\Delta x = \Delta y = \Delta z = h$

$$(\Delta\chi)_{i,j,k} \approx \frac{(\chi)i+1,j,k+(\chi)i-1,j,k+(\chi)i,j+1,k+(\chi)i,j-1,k+(\chi)i,j,k+1+(\chi)i,j,k-1-(6\chi)i,j,k}{h^2}$$

The divergence $\nabla\cdot\vec{V}$ at grid point *(i, j, k)* is:

$$(\nabla\cdot\vec{V})_{i,j,k} \approx \frac{(Vx)i+1,j,k-(Vz)i-1,j,k}{2h} + \frac{(Vy)i,j+1,k-(Vy)i,j-1,k}{2h} + \frac{(Vz)i,j,k+1-(Vz)i,j,k-1}{2h}$$

Now, the discretized Poisson equation becomes a system of linear equations:

$$(\nabla\cdot\vec{V})_{i,j,k} = \frac{(\chi)i+1,j,k+(\chi)i-1,j,k+(\chi)i,j+1,k+(\chi)i,j-1,k+(\chi)i,j,k+1+(\chi)i,j,k-1-(6\chi)i,j,k}{h^2}$$

Rearranging the terms, we get:

$$h^2\,(\nabla\cdot\vec{V})_{i,j,k} = (\chi)i+1,j,k+(\chi)i-1,j,k+(\chi)i,j+1,k+(\chi)i,j-1,k+(\chi)i,j,k+1+(\chi)i,j,k-1-(6\chi)i,j,k$$

The above equation can be represented in a matrix form as $A\chi = b$ where:

- χ is the vector of unknown values of the implicit function at the grid points.
- b is the vector containing the divergence values scaled by h^2.
- A is the sparse matrix representing the discrete Laplacian operator.

To solve the above linear system of equations, we can use numerical methods such as the Conjugate Gradient Method or Multigrid Methods.

5. **Isosurface Extraction**

The final step is to extract the Isosurface. To obtain a reconstructed surface $\partial\tilde{M}$, it is necessary to first select an isovalue and then extract the corresponding isosurface from the computed indicator function. We choose the isovalue so that the extracted surface closely approximates the positions of the input samples. We do this by evaluating χ at the sample positions and use the average of the values for isosurface extraction.

This choice of isovalue has the property that scaling χ does not change the isosurface. Thus, knowing the vector field $\vec{V}$ up to a multiplicative constant provides sufficient information for reconstructing the surface. To extract the isosurface from the indicator function, we use Marching Cubes algorithm. The Marching Cubes algorithm involves processing each voxel (a voxel is a cube with 8 vertices) and then generating the Isosurface. The voxels are classified as inside or outside the Isosurface and then the edge intersections are determined. Edge intersections are where the Isosurface intersects the edges of the voxel. Given two vertices v_1 and v_2 of an edge, with their respective function values χ (v_1) and χ (v_2), we aim to find the point p on the edge where the function χ is zero, i.e., where the isosurface intersects the edge.

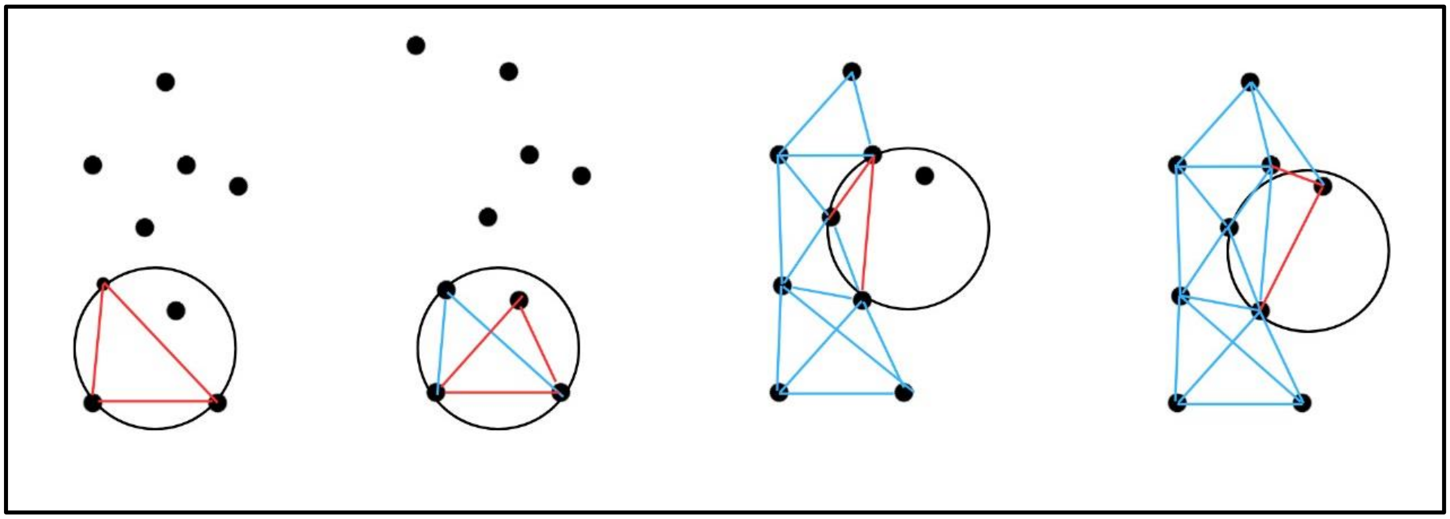
$$p = v_1 + \frac{0-\chi(v1)}{\chi(v2)-\chi(v1)}\,(v_2 - v_1)$$

The 0 in the formula represents the isovalue of the implicit function χ that we are interested in for extracting the surface. This value is typically zero, defining the surface where χ transitions from negative to positive, thus identifying the isosurface for reconstruction.

The triangles are generated using the case table and triangulation is used to connect intersection points, creating the resulting triangulated mesh of the Isosurface. After that, some post-processing could be done to improve the mesh quality. It includes Surface smoothing, Decimation, i.e., reduction of the number of triangles while preserving the overall shape and recalculation of vertex normal for better shading and visualization.

3.3 Ball Pivot Algorithm

Ball Pivoting Algorithm (BPA) is a type of algorithm in computer graphics and geometry which helps to reconstruct a 3D mesh from a point cloud. The point cloud contains all the points in 3D space and the basic work of this algorithm is to join these points to construct a 3D mesh. It constructs this mesh by considering two shapes i.e. are circles and triangles. The basic idea of BPA is to roll a ball of fixed radius over the points in the point cloud. The radius of the considered ball is calculated by choosing a starting triangle. This triangle is created using three points in the point cloud that are close to each other and later the radius of the circle is considered by fitting it inside the triangle. The radius of the ball should be chosen carefully based on the density and spacing of the points in the point cloud. Then, the ball begins to roll on one of the current triangle's edges. At this moment, the ball is touching two vertices of the triangle i.e. two points of the triangle. And while rolling through the edge, it will search for a third point that it can touch without falling off. When the ball finds the third point, it connects it with the two point of the edge and form another triangle connecting the both those points. After joining the points, it adds this new triangle to the growing mesh. And so on, it continues to use the edges of all the newly created triangles as the edge to roll the ball to find more triangles and connect them. The program will continue doing this until the ball is unable to create any more edges for a valid triangle. At that point, we can consider the surface reconstruction to be complete.



Pivoting Ball Formation. The ball of a specified radius is positioned such that it touches three points.

Triangle Formation. Joined those connecting points to form a triangle.

Iteratively forming triangles and connecting points.

The final mesh is formed.

In this process, the size of the ball's radius plays a significant role. A large radius could lead to a triangle that covers large empty spaces, potentially missing smaller details, whereas a small radius might miss connecting points that are part of the surface. This method is most effective with a point cloud that is evenly distributed. If two points are scarce or unevenly spread out, the ball may not find locations to create triangles, resulting in gaps in the three-dimensional mesh.

3.4 <u>Delaunay Mesh Triangulation</u>

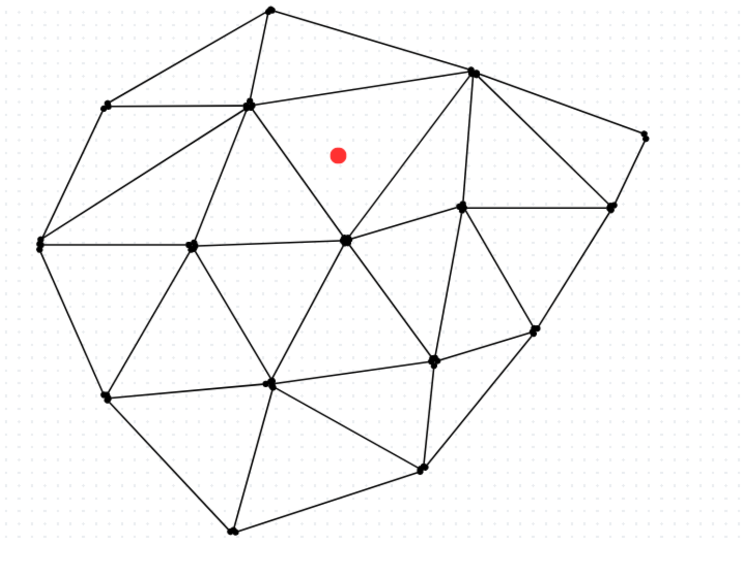
Delaunay Triangulation is a geometric algorithm used to create an optimal triangular mesh from a set of points. It's widely used in fields like geographic information systems (GIS), computer graphics, and any area requiring surface modelling and spatial data analysis.

1. **Key Concepts:**

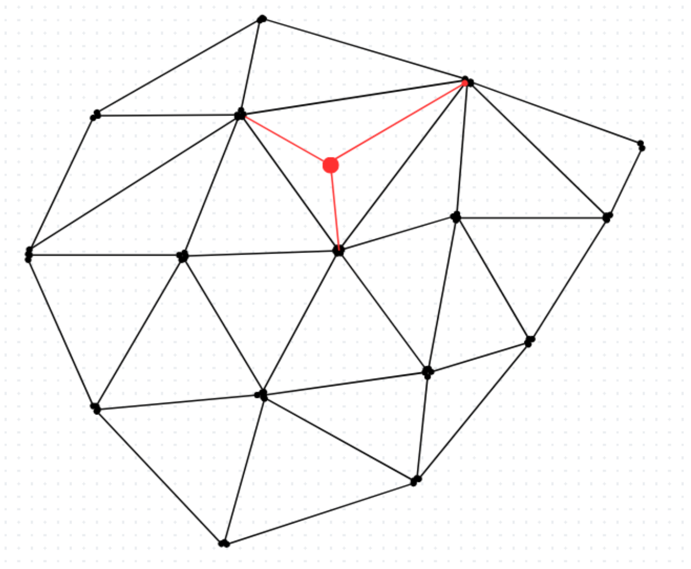
Triangular Mesh: The objective is to join a group of points to create triangles. The spatial relationships might not be accurately represented by an arbitrary triangulation.
The Delaunay Criteria: The Delaunay criterion states that no point in a triangulation should lie inside the circumcircle of any triangle in the mesh for it to be considered optimal. By doing this, the triangles are made as nearly equilateral as feasible.
Turn around: A circle that goes through each of a triangle's three vertices is called a circumcircle. Each triangle's circumcircle for Delaunay triangulation must be free of any additional set points.
Flipping the edges: To preserve the Delaunay criterion, the edges of a triangle are flipped if a point is inside its circumcircle. To achieve ideal triangulation, this operation modifies the connections.

2. Method:

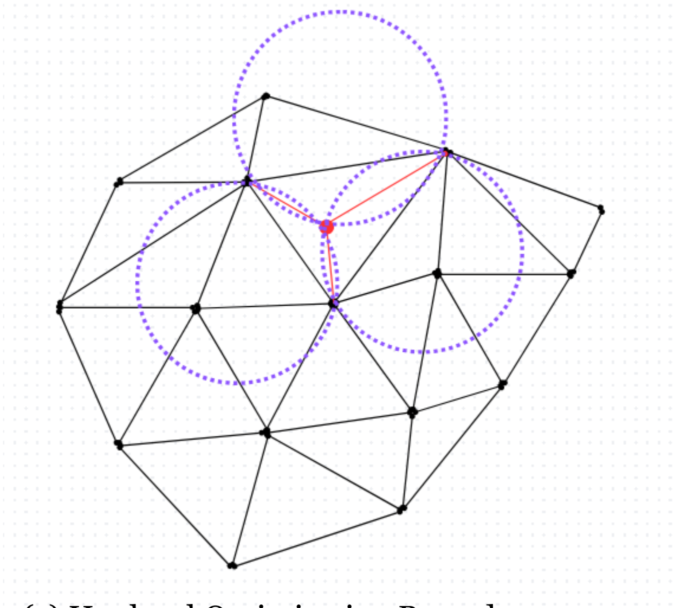
The first triangle: Start with a set of starting points that are connected by a path.
Check the circumference of a circle: Calculate the circumference of each triangle. Make sure there is no other space in the circle.
Rotate the edges: To adjust the triangle, rotate the edges if the point is inside the circle.
Secondly: Keep working until all triangles meet the Delaunay requirements.



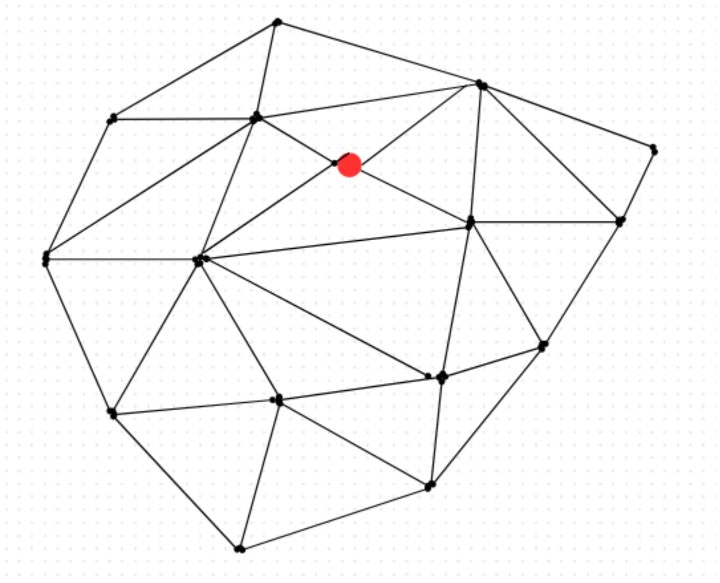
(a) Insert a new point in Delaunay triangle



(b) Connect the new point to the three surrounding vertices



(c) Use local Optimization Procedure to optimize the triangular network



(d) Generate the new Delaunay triangle

3. Advantages:

Congruent triangles: In many applications, Delaunay triangulation is better for creating triangles close to congruent triangles.
Proximity Maintenance: Keep points close to each other while maintaining their spatial relationship.
Easier screens: Screen output improves interpolation and data modelling while maintaining screen consistency.
In short, the basic technique of Delaunay computer architecture is triangulation, which provides an efficient way to combine groups of points to form a triangular surface. Its features are ideal for applications that require accurate display of the screen.

4. Formulas:

Circumcircle of a Triangle: The circumcircle of a triangle with vertices A(x1, y1), B(x2, y2), and C(x3, y3) is the circle that passes through all three vertices. The centre (xc, yc) and radius R of the circumcircle can be found using the following formulas.

The circumcentre (xc, yc) is given by:

$$xc = \frac{(x1^2 + y1^2)(y2 - y3) + (x2^2 + y^2)(y3 - y1) + (x3^2 + y3^2)(y1 - y2)}{2[x1(y2 - y3) + x2(y3 - y1) + x3(y1 - y2)]}$$

$$yc = \frac{(x1^2 + y1^2)(x3 - x2) + (x^2 + y^2)(x1 - x3) + (x3^2 + y3^2)(x2 - x1)}{2[x1(y2 - y3) + x2(y3 - y1) + x3(y1 - y2)]}$$

The radius R is given by:

$$R = \sqrt{(x1 - xc)^2 + (y1 - yc)^2}$$

Delaunay Criterion: For a triangulation to be Delaunay, the circumcircle of each triangle should not contain any other points from the set. This can be checked using the determinant method:

$$det\begin{pmatrix} x1 & y1 & x1^2 + y1^2 \\ x2 & y2 & x2^2 + y2^2 \\ x3 & y3 & x3^2 + y3^2 \\ x & y & x^2 + y^2 \end{pmatrix} > 0$$

If the determinant is positive, the point (x, y) lies outside the circumcircle of the triangle formed by the points (x1 , y1), (x2, y2), and (x3, y3).

Edge Flipping: To restore the Delaunay condition, we need to flip edges when a point falls inside the circumcircle. The flipping process involves checking if the current edge violates the Delaunay criterion and then swapping the edge with the opposite diagonal if necessary.

3.5 Heatmap Generation and Analysis

To further analyze the quality and accuracy of the 3D meshes generated from our surface reconstruction algorithms, we created depth heatmaps for each mesh. These heatmaps provided a visual representation of the depth information, displaying the distance of each point in the mesh from the camera. By translating the depth values into a color gradient, we could easily interpret the spatial distribution and depth variations across the surface.

We began by calculating the depth of each point in the 3D mesh. This involved determining the distance from the camera's origin to each vertex in the mesh. The depth values were then normalized to fit within a specific range, facilitating a consistent color mapping. We chose a color gradient that ranged from cool colors (indicating points closer to the camera) to warm colors (indicating points further away). This gradient provided an intuitive understanding of the depth variations.

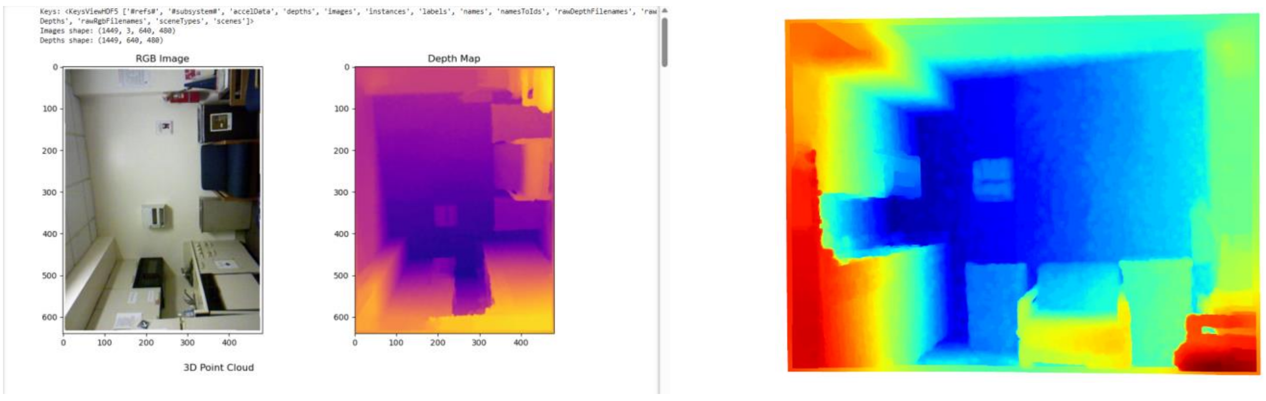
Once the depth values were normalized, we mapped these values onto the mesh vertices to generate the heatmap. Each vertex was assigned a color based on its depth, resulting in a comprehensive depth heatmap for the entire mesh. These heatmaps served as valuable tools for visually assessing the accuracy and depth information of the reconstructed surfaces. By examining the heatmaps, we could identify any discrepancies or irregularities in the depth data, thereby evaluating the fidelity of our reconstruction algorithms.

Our analysis extended to exploring the practical applications of the 3D meshes generated by these algorithms. In architectural visualization, the accurate and detailed 3D models produced by Poisson Surface Reconstruction and Ball Pivoting Algorithm proved invaluable for creating realistic and immersive representations of buildings and structures. The ability to detect and visualize defects in walls or pipes was enhanced by the precise and continuous surfaces generated by these methods, facilitating effective maintenance and repair planning.

In the field of historical artifact reconstruction, the high-quality meshes provided by all three algorithms enabled the preservation and digital reconstruction of ancient monuments and artifacts. The accurate depiction of surface details and geometries allowed for detailed analysis and virtual restoration efforts. Additionally, the uniform triangulation and structural integrity ensured by Delaunay Triangulation were particularly beneficial for creating stable and coherent models of historical objects.

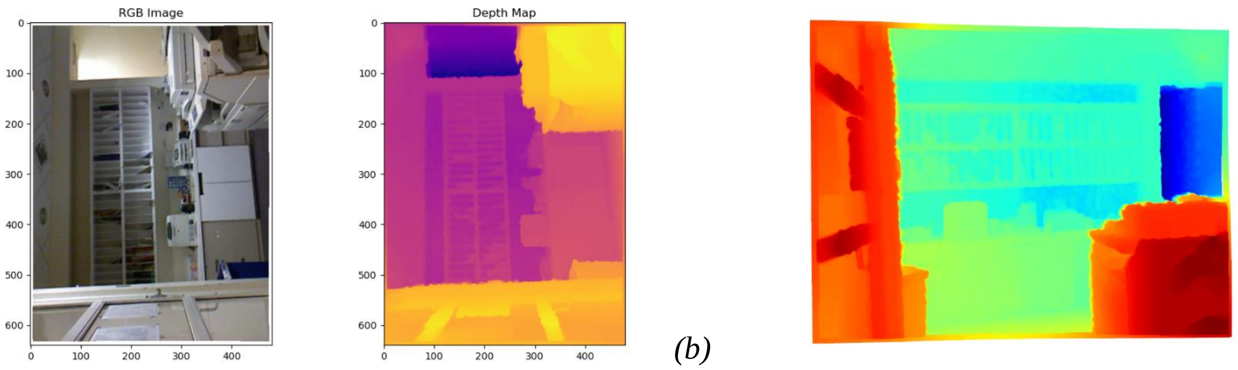
4. RESULTS AND DISCUSSION

For each 2D image given as an input in the dataset, we made a 3D heatmap of each image. Each heatmap gives the relative depth of each object seen in the image. In this case, we define an object's "depth" as the relative distance of the object from the camera. This heatmap highlights the relative depth of the objects with respect to the background, providing more clarity to the subsequent mesh where in some cases the borders of the objects may not be very distinguishable.
Figure 1 below gives the heatmaps of some of the 2D RGB images and their corresponding depth maps.

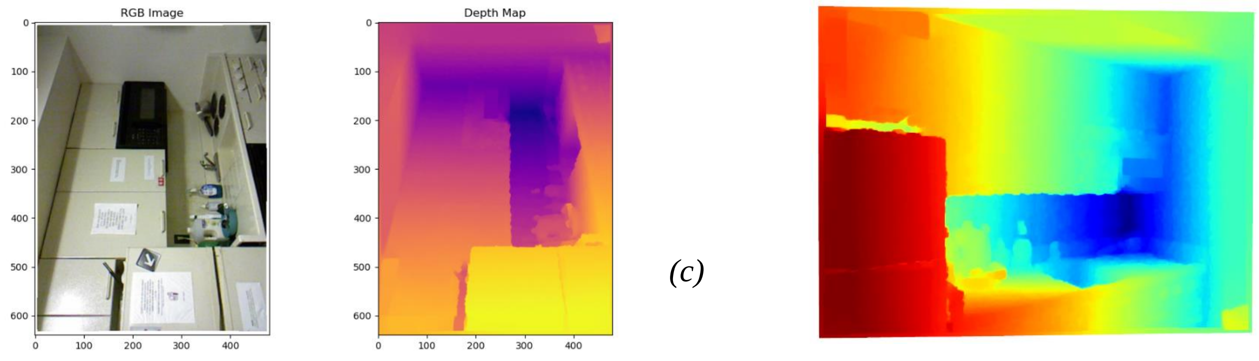


```
Keys: <KeysViewHDF5 ['#refs#', '#subsystem#', 'accelData', 'depths', 'images', 'instances', 'labels', 'names', 'namesToIds', 'rawDepthFilenames', 'raw
Depths', 'rawRgbFilenames', 'sceneTypes', 'scenes']>
Images shape: (1449, 3, 640, 480)
Depths shape: (1449, 640, 480)
```

(a)



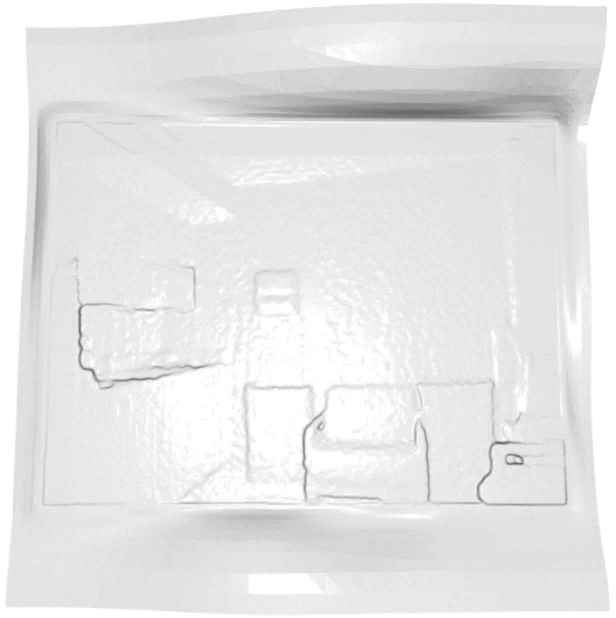
(b)



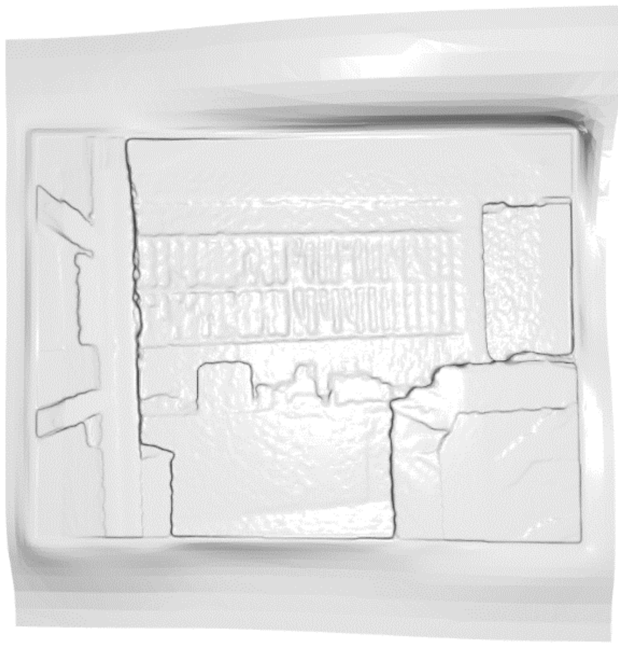
(c)

Figure 1 (a), (b) and (c) display three examples of different RGB images given as input. For each image on comparing with the depth map we see the similarities between the depth map and the generated mesh heatmap. Red colour in the heatmap has a similar purpose to yellow in the depth map, indicating objects closer to the camera. Blue/Purple represents objects that are farther away from the camera.

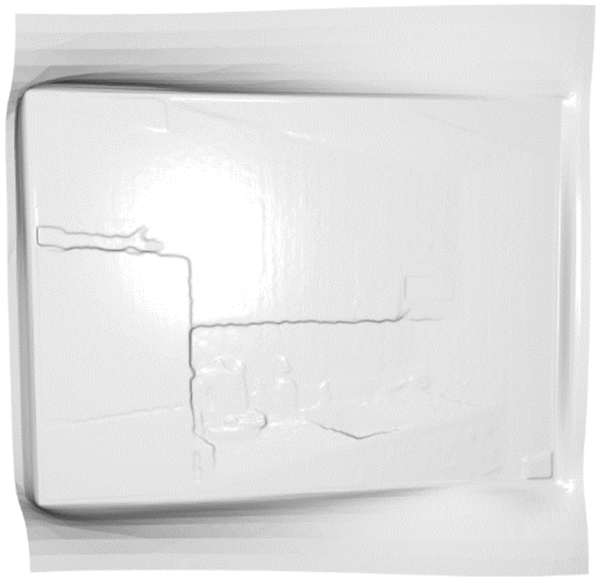
4.1 Poisson Surface Reconstruction: For our first attempt at surface reconstruction, we made use of Poisson surface reconstruction. This algorithm was able to generate a 3D mesh for the surface with a smooth background. It was able to isolate the noise (graininess of the camera image) and remove it from the resultant image. This makes sure that the 3D mesh produced was able to give an accurate representation of the surface. All details of the objects are recreated on the surface of the mesh with a good deal of accuracy and scale. We can see how the mesh is an accurate 3D representation of the original RGB image. Each mesh also shows the same depth as the heatmap. Due to the nature of the Poisson Equation, the surface is watertight, i.e., the surface is complete and has no holes in it. Figure 2 shows the generated 3D meshes of the same images as in Figure 1.



(a)



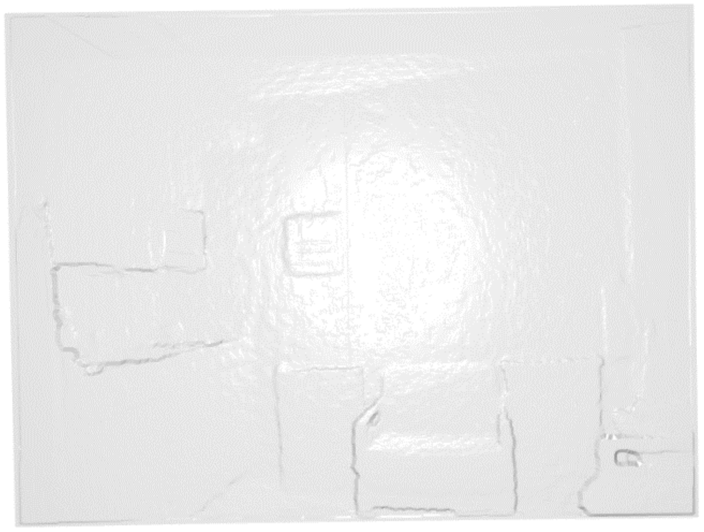
(b)



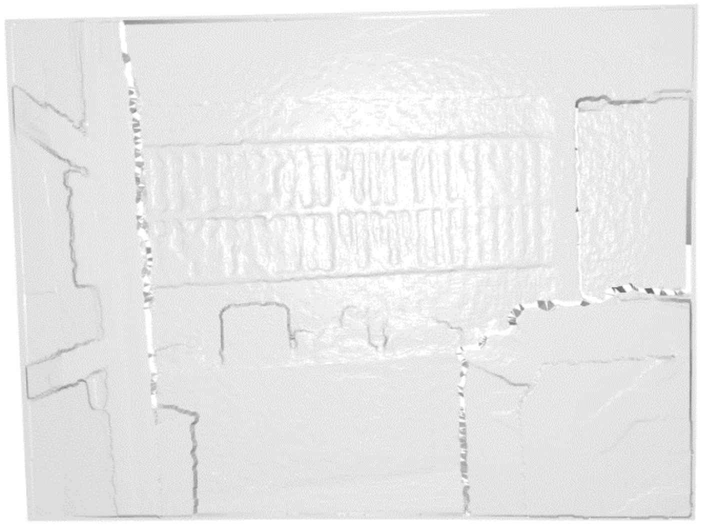
(c)

Figure 2 (a), (b) and (c) are the 3D meshes generated by the Poisson Surface Reconstruction applied on the 2D image. The mesh smooths out noise to give an accurate, watertight representation of the surface

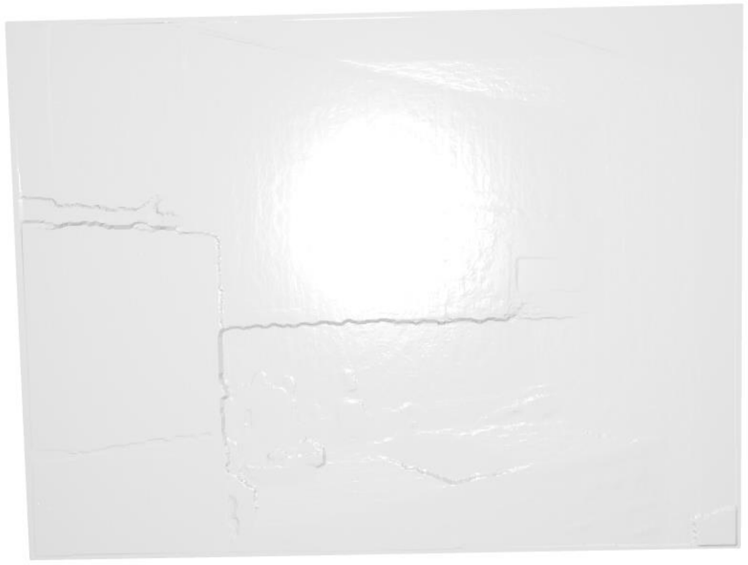
4.2 Ball Pivot Algorithm (BPA): After Poisson Surface Reconstruction, we focused on Ball Pivot Algorithm (BPA) next. BPA was able to generate a 3D model of the surface based on its corresponding 2D image**.** However, the surface generated, sometimes, had very faint borders for some objects. Some small objects were hard to distinguish from the background or other adjacent objects of similar dimensions. For this algorithm the heatmap was more important to give more clarity on each individual object in the image. Also, for certain objects, the algorithm was unable to capture the full details, which in turn led to gaps in the motion of the virtual "ball" used to map the surface. These missed points manifest themselves as small triangular "holes" on the surface and along the edges of the image. The core details of the image are still represented with decent accuracy and scale. The algorithm was adept at separating noise from the actual objects. The mesh incorporates some graininess from the original image but was able to remove a large chunk of it from the final mesh. Figure 3 showcases some of these meshes generated by BPA.



(a)



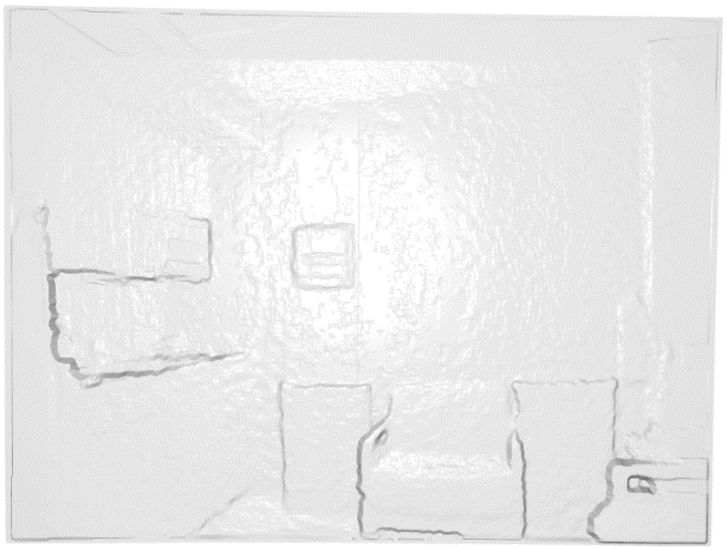
(b)



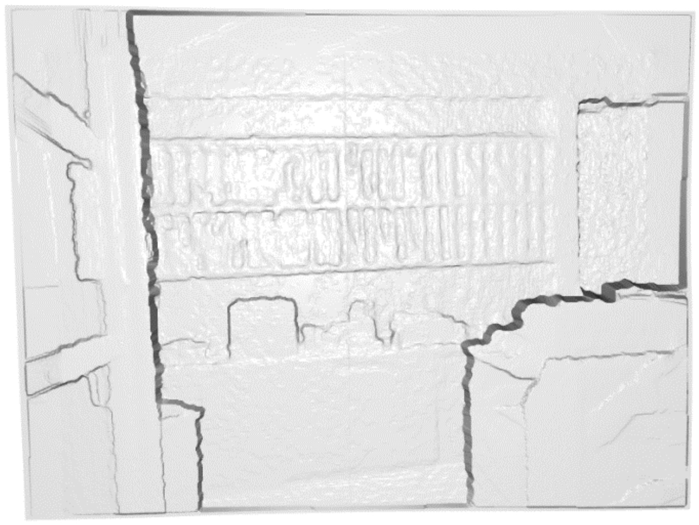
(c)

Figure 3 (a), (b), (c) shows the surfaces generated by the Ball Pivot Algorithm. The details are hard to make out for certain small objects (making it necessary to use the heatmap). The larger objects were captured with more accuracy and are rendered as such.

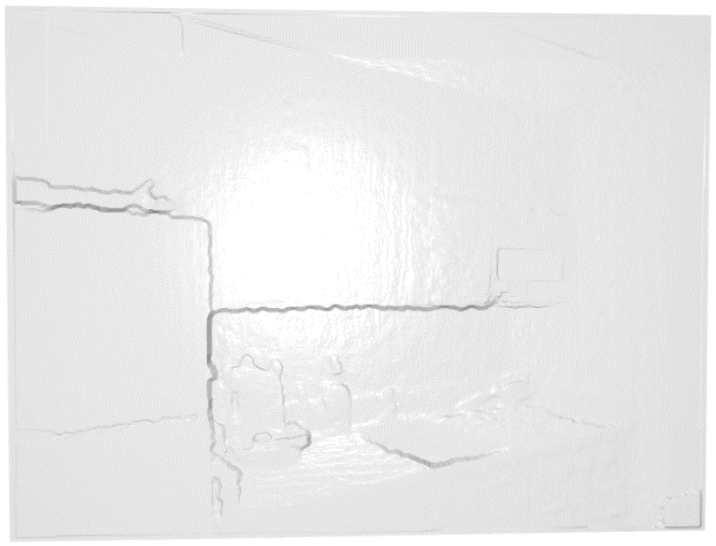
4.3 Delaunay Triangulation: This was the final algorithm we used to generate a 3D surface from the 2D RGB images given as input. The meshes generated by using Delaunay Triangulation were quite accurate and smaller objects were also detected. The borders between objects are clearer for this process and no triangulation points were missed. As a result, like Poisson Surface Reconstruction the surface is watertight and there are no holes in the mesh. The mesh versions of objects are accurate in dimension and scale with regards to the original image. However, the algorithm is unable to separate the noise from the image. The final mesh is grainier than the previous ones as the algorithm is unable to separate the inherent graininess of the camera image from the background, thus incorporating it into the final mesh. Figure 4 showcases the meshes obtained from using Delaunay Triangulation on the image.



(c)



(b)



(c)

Figure 4 (a), (b), (c) show the meshes generated by using Delaunay Triangulation on the input images. The objects are more noticeable and their boundaries are clearer. Their shapes are clearly rendered however the final mesh is grainier.

4.4 Discussion:

Analyzing the results above we can summarize the main differences between the algorithms as below:

Feature/Algorithm	**Poisson Surface Reconstruction**	**Delaunay Triangulation**	**Ball Pivot Algorithm**
Nature of the Surface	Smooth, Watertight	Triangulated mesh	Mesh following the point cloud topology
Noise Handling	Robust	Moderate	Sensitive
Normals Requirement	Necessary	Not important	Not important
Computational Complexity	High complexity due to mathematical calculations	Moderate complexity	Moderate complexity
Handling of Sparse Data	Does not adapt well to sparse data	Moderate handling of sparse data	Does not adapt well to sparse data
Handling of Dense Data	Works very well for dense data	Works very well for dense data	Works very well for dense data
Post Processing Needed	Minimal Post Processing needed	Needs Post Processing often	Sometimes Post Processing may be needed
Use Cases	High quality surface reconstruction, 3D printing	Terrain Modelling, GIS	3D scanning, reverse engineering

Ball Pivot Algorithm (BPA)

This algorithm excels in handling large and dense point clouds efficiently, generating surfaces that closely follow the topology of the input data, which is beneficial for surface reconstruction in reverse engineering and 3D scanning. It is less effective with sparse or noisy data, requires an optimal ball radius that can be data-dependent, and is not as robust in producing watertight surfaces as Poisson Surface Reconstruction.

Poisson Surface Reconstruction

It generates smooth and continuous surfaces, making it robust to noise and outliers. It produces closed surfaces even with incomplete data, ensuring high-quality outputs suitable for detailed applications. The method requires oriented normals, which is mathematically complex to execute, and it is computationally expensive due to solving a global Poisson equation. It is ideal for high-quality surface reconstruction in 3D scanning and applications requiring smooth, watertight surfaces, such as 3D modelling and printing.

Delaunay Triangulation

Well-shaped triangles are produced by this easy-to-compute method, which also ensures that no point is inside a triangle's circumcircle. It works well for generating mesh-based point cloud representations. But if the input data is not well-behaved, it may not handle noise effectively and may produce non-manifold edges or faces, necessitating post-processing to produce waterproof surfaces. This method is frequently used in mesh creation for finite element analysis, terrain modelling, geographic information systems (GIS), and as a first stage in 3D reconstruction pipelines.

5. CONCLUSION

Based on our detailed analysis of surface reconstruction techniques, we conclude that Poisson surface reconstruction is most suitable for highly sophisticated setups. This method necessitates the computation of oriented normals, which is inherently complex and computationally intensive due to the need to solve a global Poisson equation. The Poisson method excels in producing smooth and watertight surfaces, making it ideal for applications requiring high fidelity and precision, despite its high computational demands. In contrast, the Ball Pivoting Algorithm (BPA) is preferable for extensive point cloud datasets due to its efficiency. BPA constructs meshes by rolling a ball of a given radius over the point cloud, forming triangles where the ball touches three points. This method, while computationally less demanding than Poisson, struggles with sparse or noisy data and requires the careful selection of an optimal ball radius. The ball radius is crucial and must be tuned to the specific dataset, making BPA less robust in generating watertight surfaces and more sensitive to data quality variations. Delaunay triangulation serves as an effective middle ground between Poisson surface reconstruction and BPA. It is particularly adept at handling sparse datasets by constructing a tetrahedral mesh that maximizes the minimum angle of triangles, thereby avoiding slender triangles. However, Delaunay triangulation is highly sensitive to noise and requires extensive pre-processing to remove outliers and ensure data quality. Despite these challenges, it provides a good balance between computational efficiency and the ability to handle varying data densities.

The scope of this project extends to critical applications such as identifying structural cracks in building walls and pipes, medical imaging for better anatomical visualization, and the architectural reconstruction of historical monuments. These applications demand robust and accurate surface reconstruction techniques to ensure precise detection and visualization of surface irregularities.

Future work for this project could involve the integration of artificial intelligence to enhance the reconstruction process. AI can monitor input parameters, automate the pre-processing of hyperparameters, and adaptively select the most suitable reconstruction model based on the specific characteristics of the data. By leveraging adaptive learning technologies, the system can continuously improve its performance and accuracy, ultimately leading to more reliable and efficient surface reconstruction for various applications.

6. REFERENCES

1. M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson Surface Reconstruction," in Eurographics Symposium on Geometry Processing, K. Polthier and A. Sheffer, Eds., 2006.
2. F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, "The Ball-Pivoting Algorithm for Surface Reconstruction," IEEE Trans. Vis. Comput. Graph., vol. 5, no. 4, pp. 349-359, Oct.-Dec. 1999.
3. Kazhdan, Misha, et al. "Poisson surface reconstruction with envelope constraints." *Computer graphics forum*. Vol. 39. No. 5. 2020.
4. Juszczyk, J. M., Wijata, A., Czajkowska, J., Krecichwost, M., Rudzki, M., Biesok, M., ... & Pietka, E. (2020). Wound 3D geometrical feature estimation using Poisson reconstruction. IEEE Access, 9, 7894-7907.
5. Brüel-Gabrielsson, R., Ganapathi-Subramanian, V., Skraba, P., & Guibas, L. J. (2020, August). Topology-Aware Surface Reconstruction for Point Clouds. In Computer Graphics Forum (Vol. 39, No. 5, pp. 197-207).
6. Berger, M., Tagliasacchi, A., Seversky, L. M., Alliez, P., Guennebaud, G., Levine, J. A., ... & Silva, C. T. (2017, January). A survey of surface reconstruction from point clouds. In Computer graphics forum (Vol. 36, No. 1, pp. 301-329).
7. Bernardini, F., Mittleman, J., Rushmeier, H., Silva, C., & Taubin, G. (1999). The ball-pivoting algorithm for surface reconstruction. IEEE transactions on visualization and computer graphics, 5(4), 349-359.
8. Stelldinger, P. (2008, December). Topologically correct surface reconstruction using alpha shapes and relations to ball-pivoting. In 2008 19th International Conference on Pattern Recognition (pp. 1-4). IEEE.
9. An, Y., Zhao, P., Li, Z., & Shao, C. (2016). Self-adaptive polygon mesh reconstruction based on ball-pivoting algorithm. International Journal of Computer Applications in Technology, 54(1), 51-60.
10. Jiang, J. F., Zhong, Y. Q., & Zhang, Q. P. (2013). Three-dimensional garment surface reconstruction based on ball-pivoting algorithm. Advanced Materials Research, 821, 765-768.
11. Medeiros, E., Velho, L., & Lopes, W. (2004, October). Restricted BPA: Applying ball-pivoting on the plane. In Proceedings. 17th Brazilian Symposium on Computer Graphics and Image Processing (pp. 372-379). IEEE.
12. Cazals, F., & Giesen, J. (2006). Delaunay triangulation based surface reconstruction. In Effective computational geometry for curves and surfaces (pp. 231-276). Berlin, Heidelberg: Springer Berlin Heidelberg.
13. Gopi, M., Krishnan, S., & Silva, C. T. (2000, September). Surface reconstruction based on lower dimensional localized Delaunay triangulation. In Computer Graphics Forum (Vol. 19, No. 3, pp. 467-478). Oxford, UK and Boston, USA: Blackwell Publishers Ltd.
14. Thayyil, S. B., Yadav, S. K., Polthier, K., & Muthuganapathy, R. (2021). Local Delaunay-based high fidelity surface reconstruction from 3D point sets. Computer Aided Geometric Design, 86, 101973.
15. Guo, B., Wang, J., Jiang, X., Li, C., Su, B., Cui, Z., ... & Yang, C. (2020). A 3D Surface Reconstruction Method for Large-Scale Point Cloud Data. Mathematical Problems in Engineering, 2020(1), 8670151.
16. Nguyen, A. D., Choi, S., Kim, W., & Lee, S. (2019). GraphX-convolution for point cloud deformation in 2D-to-3D conversion. In Proceedings of the IEEE/CVF International conference on computer vision (pp. 8628-8637).
17. Abdulqawi, N. I. A., & Mansor, M. S. A. (2016). A computer method for generating 3D point cloud from 2D digital image. Journal of Image and Graphics, 4(2), 89-92.
18. Rakotosaona, M. J., Guerrero, P., Aigerman, N., Mitra, N. J., & Ovsjanikov, M. (2021). Learning delaunay surface elements for mesh reconstruction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 22-31).