Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

SCHOOL OF COMPUTER SCIENCE AND STATISTICS

# COMPARATIVE STUDY OF NOVEL VIEW SYNTHESIS METHODS FOR OUTDOOR ROCK WALL CLIMBING

DANIEL PREDA

APRIL 18, 2025

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
B.A. (MOD.) COMPUTER SCIENCE

# Abstract

**Abstract** There are very few tools to help both professional sport climbers and novice climbers inspect outdoor rock walls without directly climbing them. This is an issue when attempting to identify smaller holds on the wall, when the wall is inaccessible (either due to poor weather or when the climber is recovering), or for finding holds on portions of the wall that are hard to inspect visually (such as higher regions on the wall or occluded areas). One potential solution to this is to use improved computer vision and computer graphics techniques to generate a high-quality 3D reconstruction of the rock wall. 3D meshes are standard for this kind of visualisation, but often over smooth smaller geometries on the surface and tend to lose textural information, especially when looking at the model up close. This project reviews and evaluates state-of-the-art computer vision and computer graphics techniques, specifically in the domain of novel view synthesis, to find the most effective way to generate and render high-fidelity point clouds of a rock wall. This project investigates the state of the art in scanning, point-cloud acquisition, and point-cloud rendering, and compares two state-of-the-art methods for novel-view synthesis (Gaussian Splatting for Real-Time Radiance Fields, and Trilinear Point Splatting) against traditional 3D meshes using a series of quantitative metrics and qualitative evaluation. It was found from the project that although novel view synthesis is rapidly becoming more performant for real-time visualisation, its use in rock wall climbing would not be as useful as other visualisation methods.

# Acknowledgements

I would like to thank my supervisor, Anthony Ventresque, for his support and guidance in this project. I also thank my family for their love and support in everything I do.

# Contents

# 1 | Introduction - Chapter

**Introduction** Climbing is a rapidly expanding sport that is very popular worldwide, and has seen a huge increase of people participating in the sport. With this, there is an increasing need for tools to help novice climbers learn, especially for outdoor settings. Outdoor rock wall climbing has many challenges that are separate from indoor rock wall climbing. One is that usable holds on natural rock walls are more difficult to visually identify than on artificial rock walls, which have a high degree of contrast between the wall and holds that climbers can use. Variable weather and lighting conditions when climbing outdoors means that outdoor climbing is much more difficult for novice climbers. Novice climbers are often unable to complete outdoor routes with difficulty ratings that they would be able to complete on equally difficult indoor routes (1). Most climbers use 2D maps that outline a route, called "topos". They provide information about the difficulty of the route, how to access the route, and a description of the route. However, topos are very low detail, and do not provide any specific information about what holds are on the route. This makes it very difficult for climbers to know what holds to use, or if there are holds missing, especially for novice climbers (2). Moreover, professional sport climbers are often unable to practice on the route they are attempting due to poor weather, and rely on video to review their performance on sections of the climb. Relying on video for review means that the climber cannot change the view from where the video is being recorded, and may not be able to inspect particular holds on the route.

In the field of Computer Vision (CV) and Computer Graphics (CG), there has been

many developments made in 3D reconstruction and visualisation. 3D reconstruction methods like Structure from Motion (SfM) and Multi-View Stereo (MVS) allow for the creation of 3D point clouds from a series of images of an object. They have become very popular for a huge range of professional and hobbyist applications, in large part due to how easy it is to create a high quality reconstruction, from a relatively small set of photos that can be easily acquired with consumer devices. In recent years, Novel view synthesis methods have made huge advancements in model fidelity and performance. Novel view synthesis methods aim to render new views of a 3D scene from a set of input images. Recently, a set of novel view synthesis methods have been developed that produce extremely realistic renderings. Neural Radiance Fields (NeRFs) have been able to generate realistic views of 3D scenes using machine learning techniques, namely multilayer perceptrons (MLPs) (3). This method has had many developments, including increasing training and rendering performance (4, 5), as well as visual quality (4, 5, 6, 7). These methods have much higher quality texture reconstruction quality to meshes, and are approaching a similar level of performance.

With novel view synthesis methods creating extremely realistic renders at real-time performance levels, this project aims to find *if novel view synthesis could be used to create real-time, high quality visualisations of outdoor climbing rock walls*, and if so, *would these visualisations help climbers when attempting outdoor climbing routes?*

**Aims** This work will investigate the use of novel view synthesis techniques for outdoor rock wall climbing. It will first look at different point cloud acquisition techniques, namely terrestrial laser scanning (TLS), and photogrammetric techniques (SfM/MVS). It will then look at which technique is better for novel view synthesis, as well as which methods to employ when using these techniques. SfM and MVS will be employed using the COLMAP software package to create a traditional 3D reconstruction by generating a dense point cloud and mesh using the Poisson meshing algorithm. It will then investigate different metrics and their advantages

and drawbacks. It will establish a set of metrics that will be used to evaluate the state-of-the-art novel view synthesis methods. The work will then look at the current state-of-the-art in novel view synthesis techniques, namely Neural Radiance Fields (NeRFs), Gaussian Splatting (GS), and Trilinear Point Splatting (TRIPS). The metrics will then be used on the rendered results of GS and TRIPS against a set of ground-truth images, which will be obtained from the input images used in SfM and MVS. It will discuss the quantitative results, and provide a qualitative assessment of all the techniques. The work makes the following contributions :

- **A review of 3D reconstruction techniques, namely SfM and MVS, and investigates methods for data acquisition for outdoor climbing walls**
- **A detailed comparison of Gaussian Splatting for Real Time Radiance Fields and Trilinear Point Splatting, using a combination of quantitative image-based metrics (NCC, PSNR, SSIM, and LPIPS), and qualitative assessment in the context of rock wall climbing.**
- **An assessment on the use of novel view synthesis in the application of visualisation for outdoor rock wall climbing**

# 2 | Background

This section will outline the general background for novel view synthesis. This includes the methods for data acquisition, applications of 3D visualisation in other fields, and previous work done on climbing education and visualisation for climbing. It will then discuss the technical details of Structure from Motion and Multi-View Stereo to establish how camera information and point cloud data is derived for novel view synthesis. It will finally discuss the different types of metrics that are used for 3D reconstruction, and discuss the metrics that will be used for the quantitative evaluation of novel view synthesis techniques. The report will discuss the ethics of perceptual metrics and highlight the ethical concerns of 3D reconstruction for climbing.

## 2.1 3D Visualisation

3D visualisation in general refers to the process of rendering 3D scenes for a wide range of applications. In the context of this project, 3D visualisation refers to the 3D rendering of data, with the goal of making the data easier to understand or interpret. This data is often spatial data, where the visualisation simply represents the data's spatial structure in 3D, as is the case for point cloud data. 3D visualisation in this sense is used in many professional and hobbyist settings. An area that uses 3D visualisation extensively is in biomedical applications, where it is used for surface reconstruction of body parts. Harder surfaces such as the skull are particularly suitable for 3D surface visualisation and are widely used in biomedical settings (8).

3D modelling and visualisation are used in the documentation and recording of historic archaeological sites, where there is an interest in preserving fine details of the site in the 3D model. Geologists use many different scanning methods to record 3D data which are used to create digital surface models of geographical data, such as photogrammetric methods (9, 10, 11), or terrestrial laser scanning (12). These methods can then be used to render the 3D data in a realistic way, which can be used by geologists to interactively inspect the 3D data. Rocha et al. (13) investigate illustrative visualisation in the context of geological modelling, which is a method of rendering that attempts to highlight the underlying data in a clear way. Lawonn et al. (14) conducted a survey on surface-based illustrative rendering and describe illustrative rendering as "a visualization technique for measured, simulated and modelled data that [...] create more effective, more understandable and/or more communicative data representations than is possible with other visualization techniques.". The goal of illustrative rendering is to remove all unnecessary information and to clearly highlight certain parts of the underlying data. This is shown by Robb (8), where the visualisations do not attempt to be realistic and instead use simple colouring to highlight aspects of the data. Lawonn et al. (14) outline many different methods used in illustrative rendering that extract contour lines and silhouettes, and lines that "characterize particular features on the surface of the object that are not necessarily characterized by (potential) changes in visibility", called feature lines.

## 2.2 Climbing

There are many different categories of climbing that each have different goals, and importantly, different tools that are allowed to be used. One of the most popular types of climbing with professional sport climbers is redpoint. Redpoint, or a project, is when a climber will spend a substantial amount of time learning a particular route (usually several months). This is usually done for the most difficult climbs in the

world. The other types of climbing are on-site and flash, where the climber will go to a wall and climb it with minimal preparation. In on-site climbing it is seen as unethical to use any extra help, whereas extra tools are allowed in flash climbing.

For flash climbing, research has been done which investigates tools for route identification and planning. For example, Kajastila and Hämäläinen (2) create a 3D version of a "topo" which is a drawing of the different climbing routes of a given area and their difficulty. These topos are often low detail, which they argue makes longer climbs much more difficult and potentially dangerous. They surveyed their 3D topo to different climbers and found that most of the respondents said that it would make a climb easier. Many comments from respondents mention that the quality of the model is too low when up close.

A similar study conducted by Reuss et al. (15) found that their 3D visualisation was very popular in a survey they conducted, with all 23 respondents saying it was more useful than a 2D topo, and that they would use it instead. One issue with traditional meshing however is that it tends to smooth fine geometry, and also tends to lose texture information up close. There are also applications for visualisations for recreating routes. Whiting et al. (16) investigate the use of 3D modelling in creating indoor replicas of outdoor routes. They do this by creating a sparse point cloud using Structure from Motion, as well as video of a climber attempting the route on the wall. They then project the frames of the video onto the point cloud using the camera extrinsics to get the position of the holds used on the 3D model. They were then able to create an annotated model of the wall and the respective holds used by the climber in 3D.

### 2.2.1 Route reading and visualisation in rock wall climbing

In rock wall climbing, there are two different types of visualisation that climbers do in the process of executing a route. The first happens before the climb, called route

reading, where the climber will investigate and highlight important holds on the route and their type, which in turn they can use to decide what how to approach each hold, what movements must be executed and thus how to compete the route. Route visualisation refers to the process of the climber mentally visualising the route by placing themselves in the route "in their minds eye" (17). This is done before and during the climb, indicating that climbers do not simply execute the established plan, but instead approach the route with an initial plan that can change as the climb progresses (17). Climbers may take video or images of a rock wall to use as an aid for route reading, or for review of their climbing performance in the context of professional redpoint climbing, but route reading usually involves physically inspecting the rock wall with no extra visualisation tools used. Rucińska (17) investigates route reading and route visualisation in rock wall climbing through the framework of embodied and embodied cognitive science (EECS), which is a large area of research that investigates our ability to mentally visualise for the purposes of learning skills or as a "mental practice" for tasks we aim to complete (18). In it, they highlight that climbing is not a "mindless activity", where the a plan is devised that is then executed with no thought. Instead, they identify climbing as a process that involves "route-*re*-reading and *re*-visualizing of his/her strategy during the ascent". Kramer (1) investigates the transfer of climbing knowledge of novice climbers from an indoor setting to an outdoor setting. They get a group of novice climbers who then received 4 weeks of indoor climbing instruction and training. The group was then asked to attempt an outdoor climb. The group completed questionaries before and after the climb and recorded climbing the to assess their performance. Kramer (1) finds that although climbers found that indoor climbing helped them apply general climbing knowledge, such as wall body movement, engagement, and awareness of hold, they find that handholds and footholds were harder to identify visually, and that specific concepts were harder to apply in unique parts of the climb.

## 2.3 Point cloud acquisition techniques

During data acquisition, there are two main ways of collecting point cloud data, as outlined by Intwala and Magikar (19). Non-Contact methods are ones that do not contact the object and instead use indirect collection methods (e.g. images, or by using scanners) to build a point cloud. These methods require additional processing to infer the point cloud. This is achieved with a wide range of different techniques and algorithms, which will be explored in this section. Contact methods are ones that directly contact and measure the topology or shape of the object, from which a point cloud can be generated. These methods provide essentially no noise and do not require additional methods to infer the point cloud, i.e. they are direct methods . Intwala and Magikar (19) mention that for tactile methods, a predefined path for the measurement hardware is required to infer the point cloud, which would be too impractical and time-consuming for a large natural rock wall.

### 2.3.1 Terrestrial Laser Scanning

TLS uses a laser scanner that measures the distance from the scanner to a point on the target object. This is done with a known direction calibrated before scanning. From this the XYZ coordinates can be found (20) (12). Bornaz and Rinaudo (12) note that laser scanners can register millions of points very quickly. Point clouds produced by TLS are the highest quality that can be achieved through non-contact methods, however there are more difficult and less practical to setup than photogrammetric methods. TLS scanners are also much more expensive than photogrammetric methods, with budget TLS scanners costing around $5000, and high end scanners costing over $100,000. Photogrammetric methods can be achieved with any consumer RGB camera, as image quality does not drastically affect the quality of the reconstruction (11).

**Noise reduction**

Bornaz and Rinaudo (12) explain that TLS point clouds will usually have a high amount of noise, which needs to be filtered for better reconstruction. They describe a simple outlier removal algorithm, where the point cloud is quantised into meshes of a fixed size. From the meshes the median distance is found, and any points that have a larger difference from the median than the scanner are rejected. They noted that the algorithm was very effective in removing poor measurements that are acquired when the laser beam diverges and result in erroneous measurements.

**TLS Point cloud registration**

In most cases, more than one scan will be necessary to obtain the whole object. However, each scan has different coordinate spaces and must be combined into a shared reference system (12). Dong et al. (21) outline that most methods of registration begin by getting the a rough approximation of the orientation and position of points, then apply fine registration algorithms such as normal distribution transform and iterative closest point (ICP) to refine the point cloud .

### 2.3.2 Photogrammetric

Scaioni et al (9) outline two categories of mountain surfaces that each required different approaches when they were taking image data in the context of a high mountain environment. The first category includes near-flat, elongated planes that may have occlusions. These require a sequence of parallel photos, with an overlap in image features of around 75-80%. This is to ensure an average distance between common feature points of consecutive images by a set amount. They also find that they need images that are rotated by 30°-45° from the direction of the main image sequence. They also specify images that are rolled 90° from the same positions of the main sequence. In the case of occlusions, extra blocks of images have to be taken which show the geometry behind the occlusion. The other category covers large

features that have irregular characteristics that do not fall into the first category, which is less applicable to this project. There is also a discussion posed by Scaioni et al. (9) and Scaioni and Corti et al. (10) about whether it is better to use shorter baselines and more images, or less images that have longer baselines and higher resolution. Scaioni et al. (9) decided on using more images with shorter baselines to generate more points in the point cloud. Scaioni and Corti et al. (10) create two separate datasets and compare their quality. They find that more images result in a larger number of Tie points (i.e. common feature points between two images), and that the reduced image set generated much smaller point clouds (as much as 55% reduction in points). Smith et al. (11) conducted a review of SfM in physical geography and describe many practical considerations for image collection. They advise that the subject must have distinct features and texture (i.e. shiny reflective features will not be captured well), and that consistent lighting must be ensured. They also advise for large overlaps, both in terms of coverage and angles. They note that as many images as possible will help with reconstruction. They also highlight that the number of images that can be used depends on the performance of the computer running SfM. Although photogrammetry can produce results quickly, the cost of producing high-quality results is very high, making iterating quickly on results time-consuming, especially when there are a large number of input images.

## 2.4 Structure from Motion

Struture from motion is a set of algorithms that attempt to learn the external camera parameters of the input images in order to build what is called a sparse model of the object being reconstructed. The first step to this is called correspondence search. The goal of this step is to generate what is called a scene graph, which is a graph representation of the images pairs, with the images as nodes and the identified image pairs as edges.

### 2.4.1 Correspondence search

The first step in correspondence search is to find sets of local feature points for each input image. This is usually done with SIFT, as the feature points need to be invariant under geometric transformations, as it is necessary to identify the same feature points from different images. There have been more recent advances in feature recognition that learn the feature descriptions. Next, SfM looks for images that have a significant overlap in what they are capturing in the scene by using the feature set $F_i$. It does this in a brute force method, checking every image pair and looks for feature similarities between the two images, using a similarity metric for the corresponding feature descriptors. Schönberger et al. (22) note the poor time complexity of this approach $O(N_I^2 N_{F_i}^2)$, which makes SfM in general very slow for larger image sequences. This returns a set of potential image pairs, as well as their matching features. The final stage in this step is geometric verification, which attempts to find a transformation that maps the image's feature points geometrically. Outlier tolerant techniques such as RANSAC are used to establish if a transformation maps enough of the feature points between the two images. This results in a filtered set of images pairs and their feature correspondences. From this, the scene graph is constructed.

### 2.4.2 Reconstruction

Once the scene graph is constructed, SfM will use it to generate a set of pose estimates $P$ for each input image, which also called the camera poses, or scene parameters, and the reconstructed scene structure as a set of points.

### 2.4.3 Iterative Reconstruction

Schönberger et al. (22) describe the process for iterative reconstruction, which is the most popular method. In this method, an initial 'good' image pair is selected, and an initial two-view reconstruction is created. An image pair with a dense location in the image graph is essential to a good reconstruction. From this, SfM registers additional

images into the reconstruction by identifing the correspondences between images feature points and identified scene coordinates from the current set of points in the reconstruction. SfM uses RANSAC and a pose solver, to find the camera pose $P_c$ , as well as its position and orientation for uncalibrated cameras (images from which the position and orientation is unknown). Next, the registered image is checked against other registered images to identify new scene points to add to the reconstruction via triangulation. Basic triangulation involves casting two rays through the projected point from each camera matrix, then finding the intersection of these rays. However, noise always has to be accounted for in reconstruction, which complicates triangulation (23). Image registration and triangulation, and uncertainties or drift in the camera poses will affect the triangulated points, this alone will cause the reconstruction to drift and give unacceptable results (22). To fix this, a optimization algorithm called bundle adjustment is used, whereby a non-linear refinement of the camera parameters and scene points is done by minimizing the error function :

$$E = \sum_j \rho_j(||\pi(P_c, X_k) - x_j||_2^2)$$

where $\pi$ is a projection function from the scene space to image space, and $\rho_j$ is a loss function. The loss function is typically the Euclidean distance between the predicted projected point from the learned camera matrices and point positions, and the observed projected point from the original view.

### 2.4.4 Triangulation techniques

There are many different triangulation algorithms, each with their own trade-offs in efficiency and reconstruction robustness. Hartley and Sturm (23) assume Gaussian noise, and approach the problem as a minimization of the Euclidean distance of the 3D point between the rays cast through the feature point from both images. They propose a method of triangulation using a 6-order polynomial equation, which can be derived from the epipolar lines that pass through the shared feature point in both

images. The minima of this function can be found to find the most optimal point for the two image rays.

## 2.5 Multi-view Stereo

Multi-view Stereo (MVS) is a algorithm separate from SfM that uses multiple views of the same object to determine a set of 3D points that define the object. In COLMAP, MVS is used to enrich the sparse point cloud generated by SfM, and uses the learned camera parameters and corresponding images from SfM to create a dense point cloud with much more geometric detail. To do this, the image depth and normal maps must be found, which is done by drawing epipolar lines from matching pixels in corresponding views and finding where they intercept, which provides a depth estimate. It does this by describing the likelihood of a particular image *patch*(i.e. an image feature, pixel, or group of pixels), has particular depth $\theta$, given that it has a corresponding place in another source image $X_l^m$ (24). Once the depth and normal maps are computed, they are fused into a dense point cloud, which is the process of using the depth and normal maps to find dense point information. This is done by finding associated points in different images, and then taking an estimate of a fused point position and normal. Schönberger et al. (24) do this by building a directed graph of nodes, where the nodes are groups of pixels that have been filtered by their photometric and geometric quality, and the directed edges denote an association with a pixel in another image, which is determined by their depth and normal estimates. They then recursively fuse the nodes, starting with the node which contains the most pixels. The position and normal is computed by using the projection that defines the relation between the two nodes. Nodes are fused together if they have similar projected depth values, normals, and if the reprojection error is small enough. This is repeated recursively until one of the conditions are not met. A fused point is then added by taking the median of all the collected positions and normals, if there are at least 3 nodes collected. This then repeats with the next node

with the largest amount of pixels. This creates a dense point cloud that can be used in meshing algorithms like Poisson or Delaunay.

### Metrics

There are two types of metrics that are used in evaluating 3D reconstruction. The first are 3D based metrics, which will use the 3D representation and compare it against a ground-truth version. For example, a generated 3D mesh would be compared against a ground truth mesh. They are usually compared using distance-based metrics, such as Normalized Cross-Correlation (NCC) or Hausdorff Distance (25). The second type are image-based metrics, which compare the 2-D rendered result against a ground truth image. These use a wide range if different metrics, from distance based metrics such as NCC, Peak Signal to Noise ratio (PSNR), and Mean Squared Error (MSE). These are commonly used in novel view synthesis, which train on input images of the subject. Image-based metrics have the advantage of being consistent between different methods, which is difficult for 3D based metrics, as the different methods use different internal representations of 3D space.

### Image-based metrics

Normalized Cross-Correlation is an image-based metric that looks at the similarity of two images by first normalizing each image to account for lighting differences, then computing the squared distance between each pixels colour value in each image.

$$NCC(x,y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

where $\bar{x}$ $\bar{y}$ are the sample mean of each image.

Rather than pixel-wise differences, Structural similarity index measure (SSIM) looks at structural changes in the image, more akin to human visual perception. This is good for assessing overall image quality. SSIM is the weighted combination of three

comparison functions, namely:

$$l(x,y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c1}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c2}$$

$$s(x,y) = \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3}$$

$$SSIM(x,y) = l(x,y)^\alpha \cdot c(x,y)^\beta \cdot s(x,y)^\gamma$$

where $\mu_x$ sample pixel mean of image x, $\mu_y$ sample pixel mean of image y, $\sigma_x^2$ variance of x, $\sigma_y^2$ variance of y, $\sigma_{xy}$ covariance of x $c_1$, $c_2$, $c_3$ are constants, and $\alpha$, $\beta$, $\gamma$ are the weights (26). The sample mean and variance of each image is found by applying a Gaussian filter on parts of each image. This is repeated for each possible pixel position of the filter and the result is the average of all the filters' results.

Learned Perceptual Image Patch Similarity (LPIPS) is a relatively novel approach to measuring image similarities between a ground truth image and images generated by many different deep neural networks. It essentially uses a convolutional neural network (CNN) to identify and compute the distance between deep features of the source and generated images, and finds the perceptual similarity between them. Zhang et al. (27) find in their paper that first proposed LPIPS that their metric produced results that were more accurate compared to the standard metrics of PSNR and SSIM. The main issue with this is that it only works with images generated by machine learning algorithms but can work with images generated by a large array of solutions.

Peak signal-to-noise ratio Signal-to-noise ratio looks at the total noise added to a signal after processing, compared to the input signal power of an input signal. In this context, it can be used to look at the amount of image noise added to the

reconstructed image from the input signal of the initial image.

$$10log_{10}(\frac{MAX_I^2}{MSE})$$

Visual Information Fidelity - This is a metric that attempts to model the similarity of two images as a set of different random fields (sets of random variables that each take the same value), and computes the difference between the modelled random field of the source image and the distorted random field of the reference image using information theory. One interesting property of VIF is that it can model cases where the referenced image can have a superior visual fidelity to the source image. Sheikh and Bovik (28) increase the contrast of the source image and find a VIF of 1.10, indicating better perceptual quality than the source image.

$$VIF = \frac{\sum_{j \in subbands} I(\vec{C}^{N,j}; \vec{F})^{N,j} | s^{N,j})}{\sum_{j \in subbands} I(\vec{C}^{N,j}; \vec{E})^{N,j} | s^{N,j})}$$

Kerbl et al. (7) use PSNR, SSIM, and LPIPS to evaluate their results against a ground truth image, and are a standard set of metrics used in many novel view synthesis techniques (3, 4, 5, 6, 7, 29).

## 2.6 Ethical Considerations of Perceptual Metrics

Although it may not appear that there are many ethical problems with the use of perceptual metrics, there are many ethical issues with machine learning models that have only been partially explored in Computer Vision (30). Particular care must be taken with the training datasets, which are often used indiscriminately in a wide range of machine learning systems. For example, LPIPS relies on the VGG image classification network, using its deep features for comparing perceptual quality between images. However, this network is trained on ImageNet, a well known image classification dataset, which in recent years has been heavily criticised for its inconsistent and problematic labels, as well as promoting political and social biases

that were not examined when creating the dataset (30, 31). Specifically, the data set has many examples of labels that are explicitly misogynistic and racist (31). Although these labels are not often used in machine learning solutions, it is difficult to know which subsets were used in the wide range of applications that use this dataset extensively. In addition to this, Princeton University, which maintains the ImageNet dataset, has deleted most of these labels and corresponding image data, making it even more difficult to track whether these labels or images were used in a given network (31). Waelen (30) analyses ethics with regard to CV using critical analysis, which is a framework adapted from critical theory which analyses social, political, and ethical issues through power relations with the aim of emancipatory goals. One implication they find is that CV systems can cause deskilling, or a loss of skills due to the over-reliance on CV technology. This issue can be seen in the survey results collected by Kajastila and Hämäläinen (2), where many respondents dislike the more powerful tool as they make the process "too easy". Many respondents had ethical issues with the use of advanced tools for climbing on site, which usually does not allow prior knowledge of the route before attempting it. The use of novel view synthesis techniques and more traditional meshing techniques would be seen as unethical to use in flash climbing, which requires no prior information of the route, and could be seen as a tool that causes climbers to rely on the visualisation and lose the mental visualisation techniques that are central to climbing.

# 3 | Novel View Synthesis

In this chapter, the report investigates the state-of-the-art in novel view synthesis in detail. It will outline the work done in novel view synthesis, and then explore two methods in detail, namely Gaussian Splatting for Real-Time Radiance Field Rendering (GS), developed by Kerbl et al. (7), and Trilinear Point Splatting (TRIPS), developed by Franke et al. (6). The aim of this section is to establish how each technique works in detail.

## 3.1 Meshes and Novel View Synthesis

The most common method for creating a 3D visualisation today is to create a 3D mesh of the desired object. Meshes have the advantage of being ubiquitous in computer graphics and are trivial to integrate into existing projects. There are many different algorithms that generate meshes for a given point cloud. The most popular algorithm is Poisson reconstruction, which takes an oriented point cloud, and generates a surface by solving a system of Poisson equations that are generated by approximating an implicit indicator function of the desired surface (32). This generates a very good quality mesh. It captures the large geometry and the texture of the wall very well; however, Poisson meshing suffers from over smoothing of sharp surfaces, which could smooth out small holds that need to be preserved for a high-quality reconstruction for climbers. New research in 3D visualisation now aims to use custom rendering methods that render a scene directly, rather than generating a mesh. This has resulted impressive improvements in visual fidelity, however, there

are a few disadvantages. Namely, these rendering solutions are not easily integrated into existing graphics pipelines, which rely on 3D meshes extensively. They also usually require training from input image data, as well as extra information such as a sparse point cloud for GS (7), camera positions and orientations for NeRFs (3), or a dense point cloud for TRIPS (6). The quality of the extra input data is critical for reconstruction quality, which could cause inconsistent visualisation quality depending on the conditions of the site. In this section, I will explain how each method works in detail, and outline their potential issues for reconstruction of natural rock walls.

### 3.1.1 Methods of reconstruction

Once a point cloud has been generated, a visualisation can be constructed from it using many different methods. There are two broad categories of visualisation that this paper is concerned with:

#### Meshes

This is when a traditional 3D mesh is created from the point cloud, which can be integrated into 3D applications easily, and are very efficient to render. The oldest meshing algorithm was a generalisation of the convex hull problem and used a set of discs with differing radii such that their intersection contained all the input points. This can be used to generate a mesh by using sets of spheres following the same conditions (33). Another method is the ball pivot algorithm, where a "ball" is dropped on the point cloud to find three points, which is then pivoted on the mesh boundary to find new points to add to the mesh (34).

#### Novel view synthesis

This refers to a large set of methods that attempt to render the scene based on input images. Although older methods of novel view synthesis attempt to morph the input images into new views, using the camera parameters learned from SfM and point

cloud from MVS (35), modern methods use a supervised learning approach, whereby the input views are used to optimise either a set of parameters (Gaussian positions and sizes for GS), or a neural network. In the case of NerFs, the scene is modelled using a multilayer perceptron, and when the scene is being rendered, the system queries the network using ray marching through the camera to figure out what should be rendered. In the case of Gaussian splatting, the scene is represented using a set of 2-D Gaussians, with the parameters of the Gaussian being optimised to match input data and rasterised onto the screen directly. These methods have the advantage of generating higher quality textural information than meshes. Their main disadvantage is that they often take a long amount of time to train, and use very expensive rendering techniques which result in poor runtime performance. Render performance is addressed in many different ways, including rendering using a voxel grid to interpolate pixel values instead of rendering each pixel individually (29), optimizing neural network architecture (4, 5, 6), or introducing explicit 3D primitives and rasterizing the novel views instead of using ray-tracing (6, 7). The following sections will outline these methods in detail, and highlight their strengths and weaknesses for rendering rock walls.

## 3.2 NeRFs

Neural Radiance Fields are a method for novel view synthesis that was first proposed by Mildenhall et al. (3). In their paper, Mildenhall et al. (3) model a static scene as a continuous function with an input with five dimensions (3D position $(x, y, z)$ and the horizontal and vertical rotations $(\theta, \phi)$, which fully describes direction of the input view), and outputs the predicted colour and volume density $\sigma$. They are then able to render the scene by using ray marching using the following equation

$$C(r) = \int_{t_f}^{t_f} T(t)\sigma(r(t))c(r(t), d)dt : T(t) = exp\left(-\int_{t_n}^{t} \sigma(r(s))ds\right)$$

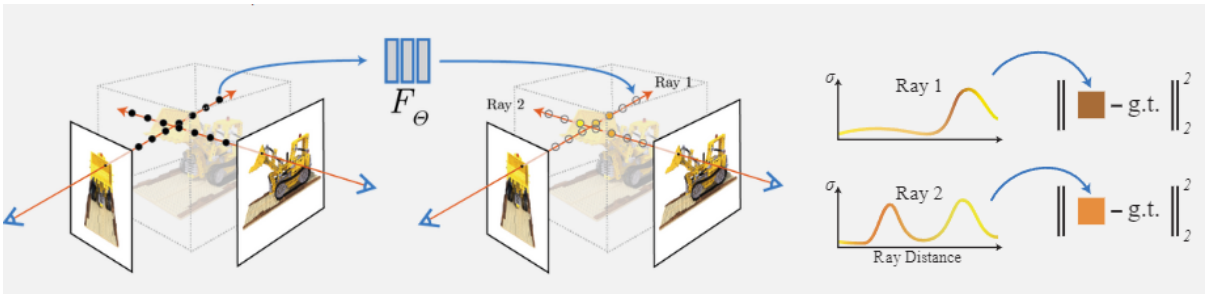


Figure 3.1: Overview of NeRFs

Where $C(r)$ is the predicted colour of the camera ray $r$, and T(t) is the transmittance of previous samples along the ray. It uses the function $\sigma(r(t))$ and $c(r(t), d)$ to sample the predicted density and colour respectfully at intervals of the camera ray. These functions are learned by sampling points from the ray and using an MLP to predict the density and colour values. To evaluate the model, NeRFs require a set of input images, as well as their corresponding camera positions and rotations. The loss function is then defined as simply the squared error between the predicted and true colour values.

### 3.2.1 Optimizations

Mildenhall et al. (3) also add two optimizations to their solution that improves the quality and performance considerably.

- Positional encoding - The model struggles capturing high frequency changes in the input data (for example, when there is a large change in colour or lighting). To help capture high frequency data, they add a positional encoding that is critical in transformers. The positional encoding maps the low dimensional input ($\mathbb{R}$) to a configurable dimensionality ($\mathbb{R}^{2L}$). In their paper, the positional encoding is used on each coordinate value and each component of the direction vector **d** separately.
- Hierarchical volume sampling - Densely sampling the network is very wasteful, as it will sample points that will not affect the final result in a meaningful way. To fix this issue, the paper trains two separate networks, a

coarse network, and a fine network. The course network is first sampled at larger intervals to establish which points contribute the most data to the final colour, this is then used to sample the fine network more densely. This greatly improves performance in training, and inference.

### 3.2.2 Problems with NeRFs

Although NeRFs produce impressive results, including at close distances to the object, the original paper is too slow for real-time visualisation. Training times are in the order of days, and rendering can take hours per frame. This is in large part due to the ray-casting rendering technique that is used to produce new viewpoints, which is expensive due to the dense sampling that is required per pixel. It also has issues with aliasing and when the input images are from varying distances (4). These issues are addressed in subsequent revisions of NeRFs in many different ways, including optimizing the hierarchical sampling by adding a separate smaller coarse network in MipNerf-360 (5), or by casting larger shapes rather than rays during rendering in Mip-Nerf (4), but performance of NeRFs still does not reach real-time visualisation, with the current state-of-the-art (MipNerf-360) taking dozens of hours for training and several seconds to render each frame.

## 3.3 Gaussian Splatting

### 3.3.1 Overview

The next big advancement that was made was with Gaussian splatting. Instead of an implicit representation that requires dense sampling, Kerbl et al. (7) propose that the radiance field can be modelled by optimizing an explicit representation, which is rendered using rasterization as opposed to ray casting techniques. They found that state-of-the-art quality could be achieved while having real time performance (over 60fps at 1080p) (7). They explain that this can be used to model radiance fields by projecting sets of 3D Gaussians onto the screen and blending their opacity ($\alpha$) values.

Instead of inferring the radiance field at every point, they create a differentiable tile rasterizer, where each tile is 16x16 pixels. The explicit representation of the radiance field eliminates the need for expensive dense sampling, and can be more directly rendered using rasterization. GS takes a set of images as input, with corresponding camera data that describes the image (i.e. the camera's orientation, and intrinsic parameters), which is generated using SfM. Gaussians are created and are initialized to the sparse point cloud from SfM. The Gaussians are defined by position x, covariance matrix $\Sigma$, and opacity $\alpha$. The view dependent colour appearance of the radiance field is represented using Spherical Harmonics. The Gaussians are modelled using the following formula:

$$G(x) = e^{-\frac{1}{2}(x)^T\Sigma^{-1}(x)}$$

Where x is the position and $\Sigma$ is the Gaussians covariance matrix. They also have an opacity value that is used in blending. These Gaussians can then be projected from the 3D world space to the screen space using the following projection matrix:

$$\Sigma' = JW\Sigma W^T J^T$$

Where $W$ is the camera projection matrix and J is the projection matrix. The authors further break down the covariance matrix into a product of the Gaussians scale and rotation:

$$\Sigma = RSS^T R^T$$

Where $S$ is the scaling matrix and $R$ is the rotation matrix. This is done to ensure that the covariance matrices maintain valid physical values during optimization, by indirectly optimizing scale and rotation which cannot devolve to invalid values during training. The derivatives of these functions are defined explicitly, instead of finding the derivative automatically at runtime, which massively improves training performance.

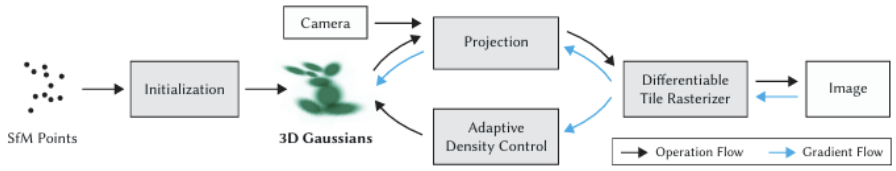


Figure 3.2: Overview of Gaussian Splatting

### 3.3.2 Optimization

Kerbl et al. (7) use stochastic gradient descent for optimisation, which optimizes a function by evaluating the gradient of that function and the cost function at randomly sampled points. Unlike normal gradient descent, this does not require evaluating the whole dataset, making optimization much more efficient. This is used to optimise the Gaussian covariance matrices through their scales and rotations, which are represented as a scaling vector and quaternion, respectively. The loss function is calculated by iteratively rendering the set of Gaussians from each camera viewpoint from the input and comparing it against its corresponding input image. The loss function is the combination of the L1 score with D-SSIM, and a learning rate term $\lambda$:

$$L = (1 - \lambda)L_1 + L_{D-SSIM}$$

### 3.3.3 Adaptive Density Control

The authors explain that their solution requires the destruction and creation of geometry, that is, it must be able to delete Gaussians that over-represent or under-represent geometry, as well as the ability to optimize how they are placed in the scene. In order to achieve this, they employ what they call density control steps in the iterations of training. Every 100 iterations after a "warm-up" optimization, they add or prune Gaussians according to several conditions. They firstly remove Gaussians that have a very low opacity, as they do not contribute anything to the scene. They then attempt to find under or over constructed areas of geometry in the scene. This is done by looking for high position gradients, which they observe

Figure 3.3: Example of GS render, ground truth on left

corresponds to Gaussians that are either covering overly large areas of the scene (over-constructed), or are not covering geometry well (under-constructed). If the positional gradient of a Gaussian is over a certain threshold $\tau_{pos}$, the area is "densified". In the case of under constructed regions, a copy of the Gaussian is created and moved in the direction of the gradient $\tau_{pos}$. For over-reconstructed areas, the large Gaussian is split into two, which their scales reduced by a factor $\phi = 1.6$. They then determine the position of the Gaussians by sampling the original 3D Gaussian that was split by using it as a PDF.

### 3.3.4 Rasterization

The key to the high performance of GS for Radiance Fields is its differentiable tile rasterizer. Their rasterizer is able to pre-sort Gaussians for the whole image at once, rather than having to sort per-pixel. It is also able to propogate gradients over any number of Gaussians that are blended, with only a constant memory overhead per pixel. It does this by firstly splitting the screen into 16x16 tiles. It then culls Gaussians that have do not have a 99% confidence interval intersection with the view frustum, similar to face culling in traditional mesh rendering pipelines. They also cull Gaussians with extreme positions, such as when they are too close to the near plane or very far from the view frustum. For each Gaussian, they assign a key that combines an ID of the tile it occupies, as well as its depth in view space. This is then used to sort the Gaussians using a fast radix sort on the GPU. From this, a sorted list

of Gaussians can be generated per tile by identifying the Gaussians with the highest and lowest depths with the same tile ID. For rasterization, a block of threads are run on the GPU for each tile. Then many blocks load the Gaussians into GPU memory, and then accumulates the colour and $\alpha$ values for each pixel in the tile by iterating through the list front-to-back. The thread that is accumulating these values stops with the accumulated opacity for the pixel reaches 1. For the backward pass (i.e. back-propagation of the gradients for learning), they did not restrain the number of blended primitives that can receive gradients for learning. This means that they must keep track of every accumulated blended point that occured during the forward pass. They do this by traversing the tile lists again, this time moving in the opposite direction, and recovering the opacities of each Gaussian by dividing the total accumulated opacity that was stored at the end of the forward pass and dividing by each points $\alpha$. These are then used as the coefficients required for the gradient computation.

### 3.3.5 Benefits and Drawbacks

GS can produce very high quality views with training times that are faster than most methods (30 minutes). Due to its compact representation of the 3D scene, as well as a highly optimized rasterizer, GS can reach interactive frame rates at 1080p. The main issue with GS is that the representation can break down when zooming in on the model, and struggles capturing textural and geometric information, due to the use of a sparse point cloud.

## 3.4 Trilinear Point Splatting

### 3.4.1 Overview

Franke et al. (6) propose a method for real-time neural rendering that combine advantages of GS, while producing texture quality that can exceed the quality of points clouds and GS, called Trilinear Point Splatting (TRIPS). It does this by

splatting the points from a dense point cloud input trilinearly onto an image pyramid. The image pyramid is turned into a feature pyramid and passed through a gated CNN, the output is then run through a spherical harmonics and tone mapper module to create the final image.

### 3.4.2 Trilinear Splatting

For each point in the dense point cloud, TRIPS projects the point and its size into screen space. It then uses the point size to determine the resolution layers in the image pyramid to splat the point. It then renders to those layers as a 2x2 splat. This effectively allows for different splat sizes similar to the method used in GS, even though it is only rendering constant sized splats. This is because the splats made to lower resolution layers contribute larger, low frequency features when they are upsampled, and splats made to high resolution layers contribute high frequency or smaller features to the final image. The alpha value of the splatted pixels is recomputed into a new alpha value $\gamma$ using the following equations:

$$\gamma = \beta \cdot \iota \cdot \alpha$$

$$\beta = (1 - |x - x_i|) \cdot (1 - |y - y_i|)$$

$$\iota = \begin{cases} 1 - \frac{|s - s_i|}{2^L_{upper} - 2^L_{lower}} & s \geq 1 \\ \epsilon + (1 + \epsilon)s & s_i = 0 \wedge s < 1 \end{cases}$$

Where $\beta$ is the bilinear weight within the image, and $\iota$ is the linear weight between layers. If the point size $s$ is less than one, then the point is very far away in screen space. The second case for $\iota$ ensures that the point is not lost. It is also added to the first layer of the image pyramid. Each pixel in the image pyramid keeps a list of up to 16 splatted fragments that contribute to the pixel value, which are then combined into a final colour value using front-to-back alpha blending. The result is a feature pyramid, where each pixel is the combined colour information of up 16 fragments.

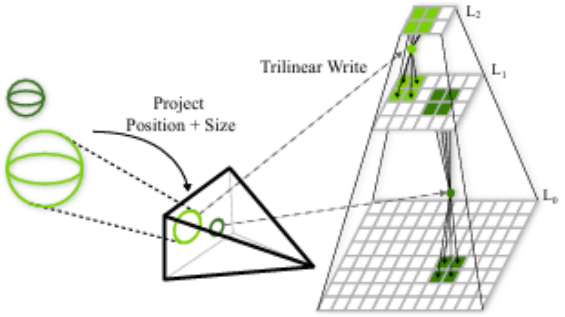


Figure 3.4: Trilinear splatting interpolates between two layers of the image pyramid

### 3.4.3 Neural network architecture

Each layer of the feature pyramid is fed into a small CNN. The architecture of the CNN contains a single gated convolution layer, a skip connection layer, and a concatenation layer, which takes the output of the previous layer and combines it with the input of the next layer. A gated convolution layer is one where a mask is added that tells the network to ignore certain input features. This is crucial to the network architecture as we do not want to change the parts of the image that have already been written by previous layers of the network. The output is then upsampled and fed to the next layer of the feature pyramid. The authors mention that this resembles a decoder network, due to the small number of features, convolutional layers, as well as the small amount of pixels. This makes this network very small and easy to train. As well as this, because the trilinear splatting reduced the number of large holes considerably, the model can focus its small number of features optimizing the texture reconstruction, as opposed to hole filling that made previous solutions require larger networks. Gated convolutions ensure that other layers of the network do not affect information that was filled in by previous layers of the network. It does this by adding a learnable soft mask that is then applied on the input layer with a non linearity added, usually sigmoid function (36).

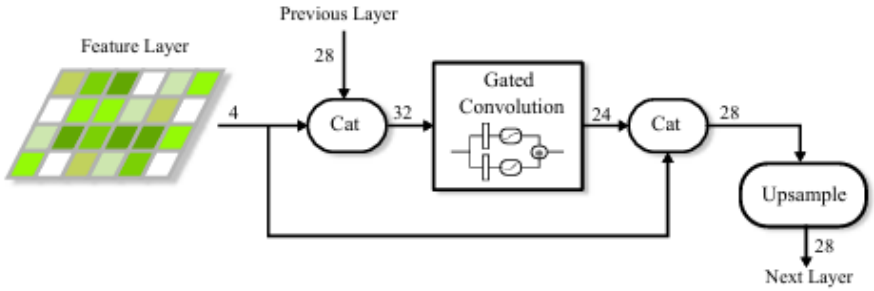


Figure 3.5: TRIPS model architecture

### 3.4.4 Gated Convolutions

Yu et al. (36) investigate the use of gated convolutions in the context of image inpainting, and find that traditional convolutional layers perform poorly as they cannot differentiate pixels or regions that are valid (i.e. the network should affect their values), or already edited or affected pixels that we do not want to change. They find this results in unwanted artefacts in the output results. They address this problem by adding a learnable soft masking layer on the output of the feature layer. This convolution layer learns to identify what channels and positions in the image to focus on. In TRIPS, this is used to learn which regions of the image to focus on at each layer, as much of the upsampled image from the previous layer has information from the previous splats that should not be changed as much as the empty regions in the image.

### 3.4.5 Rendering

The renderer used in TRIPS has a similar goal of blending splats per pixel, but approaches it differently than GS. Each point is firstly projected in the view space, where its view space position and point size is stored in a buffer, which also counts how many points are mapped to each pixel. This count is used to index into a contiguous array that is used for all layers of the image pyramid. Splatting is then done, and a duplicate of each point is stored along with its z value, and an index to the points information in each pixel list. Then each pixel list is sorted and

Figure 3.6: Example of TRIPS render, ground truth on left

accumulated as described in the previous section. For performance reasons, the pixel lists are constrained to at most 16 elements. Although they note that it did not affect the quality of results, this differs from GS, which is unconstrained in the number of per-pixel primitives for blending. The pixel lists are then sorted using bitonic mergesort on groups of 32 threads on the GPU. The sorted lists are then stored for the backward pass, which the authors say allows for fast back-propagation during training.

### 3.4.6 Optimization

For training, the model is compared against the input images used in SfM. The model then optimises the point positions, sizes, features (alpha values and depth), the camera parameters and poses, neural network weights and the parameters used for the physical-based tone mapper. To evaluate loss, they use a perceptual loss function that is obtained by running the output and reference images through an image classification network (VGG). This is from work by Johnson et al. (37), where they evaluate a style transfer and super-resolution network with the same perceptual loss function. Similar to LPIPS, they compare the deep features of the classification network. However, Franke et al. (6) use the same loss function from GS for the first 50 epochs, as the benefits of VGG-loss are minimal and is a very expensive loss function.

### 3.4.7 Benefits and Drawbacks

Due to the minimal neural network, as well as the rasterised point splatting techniques, TRIPS has faster training times than NeRFs and other neural rendering techniques (around 2-4 hours in the paper, around 6 hours on an NVIDIA RTX A4000), with real-time performance. It also produces results that preserve high quality texture and geometry, which will be discussed in the evaluation of techniques.

# 4 Method

In this section, the report outlines the approach and comparison of the different rendering techniques. It will first discuss the data acquisition, including what equipment was used, where and how the image data was collected. The setup of the three separate rendering methods that will be compared will be outlined.

## 4.1 Data collection

I decided to use my own image dataset of an outdoor rock wall. I chose to do this because there were not many existing SfM or TLS datasets that were available, especially image datasets. It was not practical to obtain a TLS scanner to collect TLS data, and thus an image dataset was collected instead. This was done to investigate image acquisition techniques that would result in the best reconstruction from SfM and MVS. Although this was investigated in by Scaioni et al. (10), it was applied to a different context, and did not cover the quality of mesh reconstruction. I aim to find whether these image capture techniques help in reconstruction for rock wall climbing specifically. I chose to gather image data from an area in Dalkey Quarry in Co.Dublin. Dalkey Quarry has many rock walls that are easily accessible and are also often visited by hobbyist climbers. For these reasons, it was a good area to collect image data. I initially went to the area and found three different walls that I wanted to create scans of. I then collected image data of each wall using three different devices, namely a consumer smart phone, a DLSR camera (Canon EOS 250d), and the Azure Kinect, which is a now deprecated 4K camera with depth sensors developed

by Microsoft in 2020. I then took images of each wall with the following method, as outlined by Scaioni et al. (9). They prescribe taking many images with short image to image baselines (that is, the distance between each subsequent image), and with high overlap (70-80%). Every second to third image, an extra image is taken with the camera panned by 15-30 degrees, and every 4-5 images an image is taken with the camera rotated 90 degrees. They explain that this is to help SfM learn camera parameters, as the algorithm can easily differentiate large changes in camera rotation. This method was repeated for all devices and walls. I also ensured that I took the images on a day with consistent lighting with uniform lighting, which is advised by Smith et al. (11). When dealing with occlusions on the rock wall, extra blocks of images were taken, using the same technique, but starting at different positions and angles to the subject. After collecting the datasets, 70 images for the first two sites on both devices, and 124 images of the third site on both devices.

### 4.1.1 Constructing the point cloud

Using this dataset, I constructed sparse and dense point clouds of each wall using both the consumer phone camera, and the DLSR. I did not use the data from the Azure Kinect, as it failed to produce adequate depth maps at the distance required for SfM. To create the point cloud, I used the COLMAP software package, developed by Schönberger et al. (22), which includes methods for SfM as well as the ability to create dense point clouds using MVS. I used the "High" quality present for SfM, and kept all parameter defaults for all reconstructions. I then ran the dense reconstruction on those outputs, which generates depth and normal maps from all images, and fuses them to create a dense point cloud. I also used the parameter defaults for all dense reconstructions. I finally created meshes of all the results using COLMAP's inbuilt mesher, which uses a Poisson meshing algorithm. Again I used system defaults for all meshes.

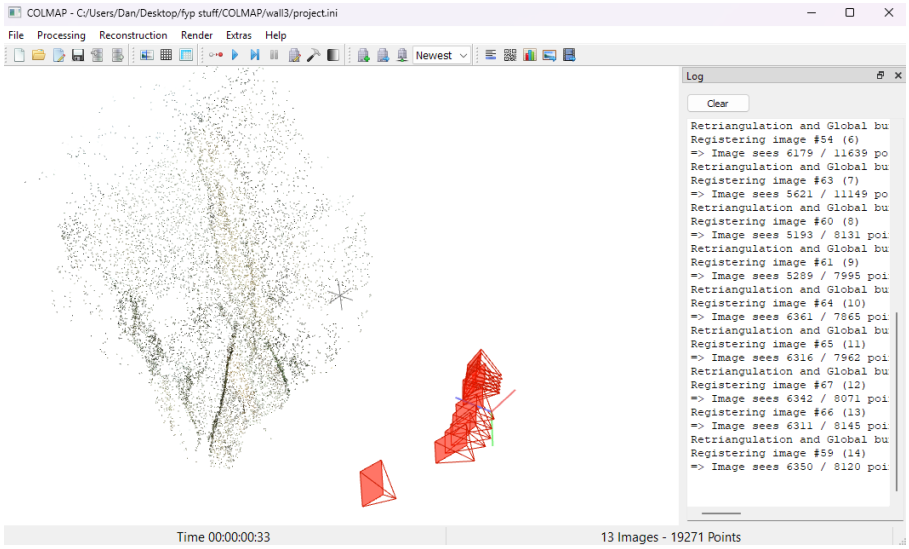


Figure 4.1: Finding camera positions and sparse point cloud using COLMAP SfM

### 4.1.2 Point cloud results

From the results, I found that the exact amount of image overlap and camera rotations were not as important for reconstruction quality as the amount of images collected. The first two walls were well reconstructed, but had noticeable gaps in the point cloud, which resulted in holes in the generated mesh. Both these walls had around 70 images each. The third wall did not suffer from this, and had excellent reconstruction results, although there was some areas of the mesh that were over-constructed from MVS. The third wall had 124 images collected, indicating that the sheer number of images is more important than how consistent their overlaps are, or how many calibration images are included. From these results, I decided to only use the third mesh and sparse reconstruction for the neural rendering methods in the following section.

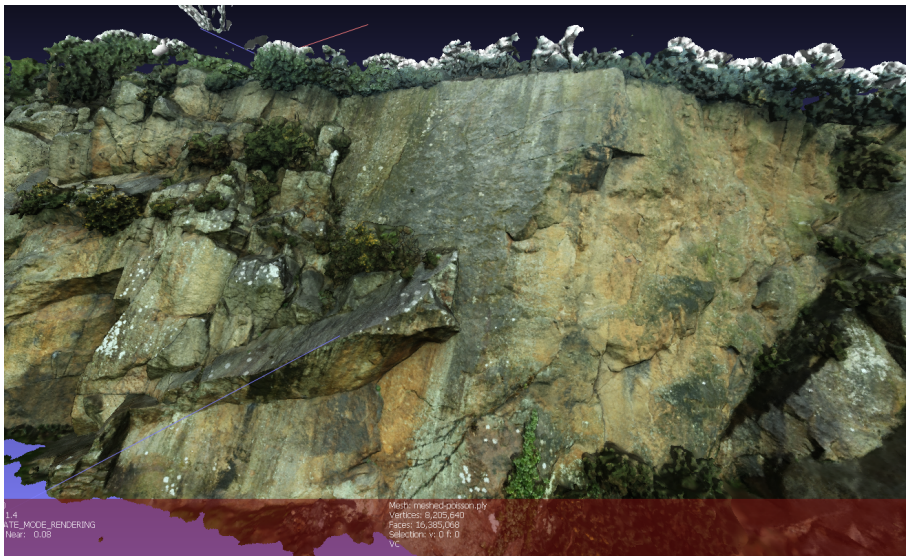

Figure 4.2: Reconstructed 3D mesh of rock wall using SfM and MVS in MeshLab software (38)

## 4.2 Metrics

I decided to use image-based metrics, as it was more practical for me to get ground truth data for comparison. As well as this, each rendering technique has completely different internal representations. In the case of GS, it is the set of 3D gaussians that describe the scene, whereas for TRIPS, it is the feature pyramid containing pixel splats. Comparing these implementations would not give any insight into how much better or worse they are performing in comparison to each other. I decided to use three different traditional pixel-based metrics, SSIM, PSNR, and MSE to as baseline metrics. These metrics are not as sophisticated as perceptual metrics, but they provide a good idea of how the visualisations capture the general scene, as well as structural features in the case of SSIM. I also included a perceptual metric, namely LPIPS. Despite the ethical issues with LPIPS specifically, it it used in the original papers for GS and TRIPS, and does not directly use the labels from the ImageNet dataset, so I felt that it would be fine to include to verify that the reconstruction produced similar quality to that of the results in the original papers. The mesh was not evaluated as it was not practical to take viewpoints at the different image positions, it is also clear that it would perform far worse than GS and TRIPS, and

would not provide any insight into its visualisation performance.

### 4.2.1 Implementation

I wrote a Python script that implements each of the metrics that I could use to evaluate the rendering results and also create graphs for comparison. I used the original implementation of LPIPS created by Zhang et al. (27). I used the PyTorch implementations of SSIM, PSNR, and NCC. LPIPS used the weights from VGG image classification network. PSNR, SSIM, and LPIPS metrics take RGB tensors as input, and NCC took the same tensor flattened to a 1-D array, all values were normalized to the range [0-1] as floating point values. Finally, I used matplotlib to generate charts of the metric results.

## 4.3 Rendering Techniques

I decided to use the implementation for Gaussian splatting written by Kerbl et al. (7) and TRIPS by Franke et al. (6). These techniques are the current state of the art in novel view rendering quality, with both approaches achieving real time performance. In the following sections, I will outline how I setup each project to run the rock wall dataset, including any changes I made to run the solutions on my system.

### 4.3.1 Gaussian Splatting

I used the implementation written by the authors at the github repository at https://github.com/graphdeco-inria/gaussian-splatting. In order to run the project, I followed the instructions from the repository. I installed the NVIDIA CUDA Toolkit, version 11.0.8, and Visual Studio 2022, version 17.2.23 in order to build the project. This was needed as the latest version of Visual Studio is not compatible with older versions of CUDA. I then used the trainer to train a set of Gaussians on the sparse point cloud dataset of the rock wall. Because the trainer does not support certain camera models, I needed generate a new point cloud with a different camera

model. The trainer did not need any changes to its hyperparameters, and was evaluated to the full 30,000 iterations that was achieved in the original paper.

### 4.3.2 TRIPS

I again used the implementation from the authors of the paper. The project was built from source using the same versions of CUDA and Visual Studio used for GS. However, the version of Visual Studio used was unable to add all the dynamic libraries required and had to be added manually to the build. In order to run the model, the COLMAP dataset was converted to the format used by TRIPS using the authors c++ script, which I edited minimally so that it was compatible with Windows commands. Due to memory constraints, the resolution of the training images was downscaled to 2000x3000 (by using the render_scale parameter set to 0.5). When training the model, some of the model parameters had to be changed again due to memory constraints on the GPU. Namely, the outer batch size was reduced to 1, and inner batch size was reduced to 2. This should theoretically affect the result quality, but I found the model results satisfactory with these parameters. All other parameters was set to their defaults.

# 5 | Results and Discussion

This chapter will present the quantitative results of the two methods, and compare the differences in visualisation performance. It will then explore their quality for rock wall climbing specifically in the qualitative assessment, where the close-up fidelity of the visualisation will be discussed. Finally, it will discuss whether these visualisations are useful for the domain of outdoor rock wall climbing, drawing on the work explored in section 2.2.

## 5.1 Quantitative Results

Metrics were ran on 29 different images from the original dataset on both TRIPS and GS, and generated the following results. GS scored extremely high on pixel based metrics, reaching 0.965 average for NCC, and 25.5 on PSNR, with a very small variation in score for NCC, with more variation for PSNR (single worse image score for NCC was 0.94, PSNR was 22.21). GS also performed very well on LPIPS, with an average score of 0.286, but with a higher scale of variability (single worse score was 0.355, best was 0.20, note that a lower LPIPS score indicates less perceptual difference between the images). From the more traditional pixel-based results (SSIM, PSNR, and NCC), GS performs far better that TRIPS, with the average score of GS scoring around 20-30% better per image than the TRIPS render, with an average NCC of 0.709 and very poor PSNR scores, with an average of 16.6, with a single worse score of 11.36. TRIPS performs best in LPIPS, with an average of 0.33, with the a single worse performance of 0.49, and best score of 0.23. This is mainly in part because

| | NCC ↑ | SSIM ↑ | PSNR ↑ | LPIPS ↓ |
|---|---|---|---|---|
| TRIPS | 0.710 | 0.580 | 16.638 | 0.331 |
| GS | 0.966 | 0.727 | 25.525 | 0.287 |

Table 5.1: Average of all quantitative results

TRIPS uses VGG loss as its loss function for training, meaning there is a bias toward perceptual metrics. The main reason for the difference in performance the other metrics is due to the smooth featureless parts of the image that GS can more compactly represent with large Gaussians. In comparison, TRIPS needs to approximate large areas with little texture with many 2x2 pixel splats. Although they addressed this using the image pyramid to model low-frequency information in the render, it is not as effective as the approach made by GS. The TRIPS renders are also much higher resolution in comparison to GS renders, with TRIPS rendering 2000x3000 images, whereas GS is rendering 1080p images. The lower resolution of GS images makes its performance in pixel-based metrics more inflated than TRIPS. Nonetheless, the large lower frequency regions of the images are captured in GS with higher fidelity that TRIPS. However, in perceptual metrics, TRIPS is more competitive with GS. TRIPS scores only marginally worse than GS in most of the images tested. As is mentioned by the authors, this is largely because TRIPS uses perceptual metrics as the loss function for training, namely the using a loss from VGG, which is the same model that is used in LPIPS. Although this metric is more biased to LPIPS as result, it still indicates that LPIPS performs well in rendering high quality novel view-points. Again, the metric performs worse than GS due to the large textureless areas that GS can capture more compactly with less optimisation than TRIPS. Another reason for the poor training quality is the choice of parameters for TRIPS, which had to be lower due to GPU memory constraints. In comparison, GS did not require any changes to its parameters, which allowed it to achieve higher quality that more closely matches the findings of the original paper.

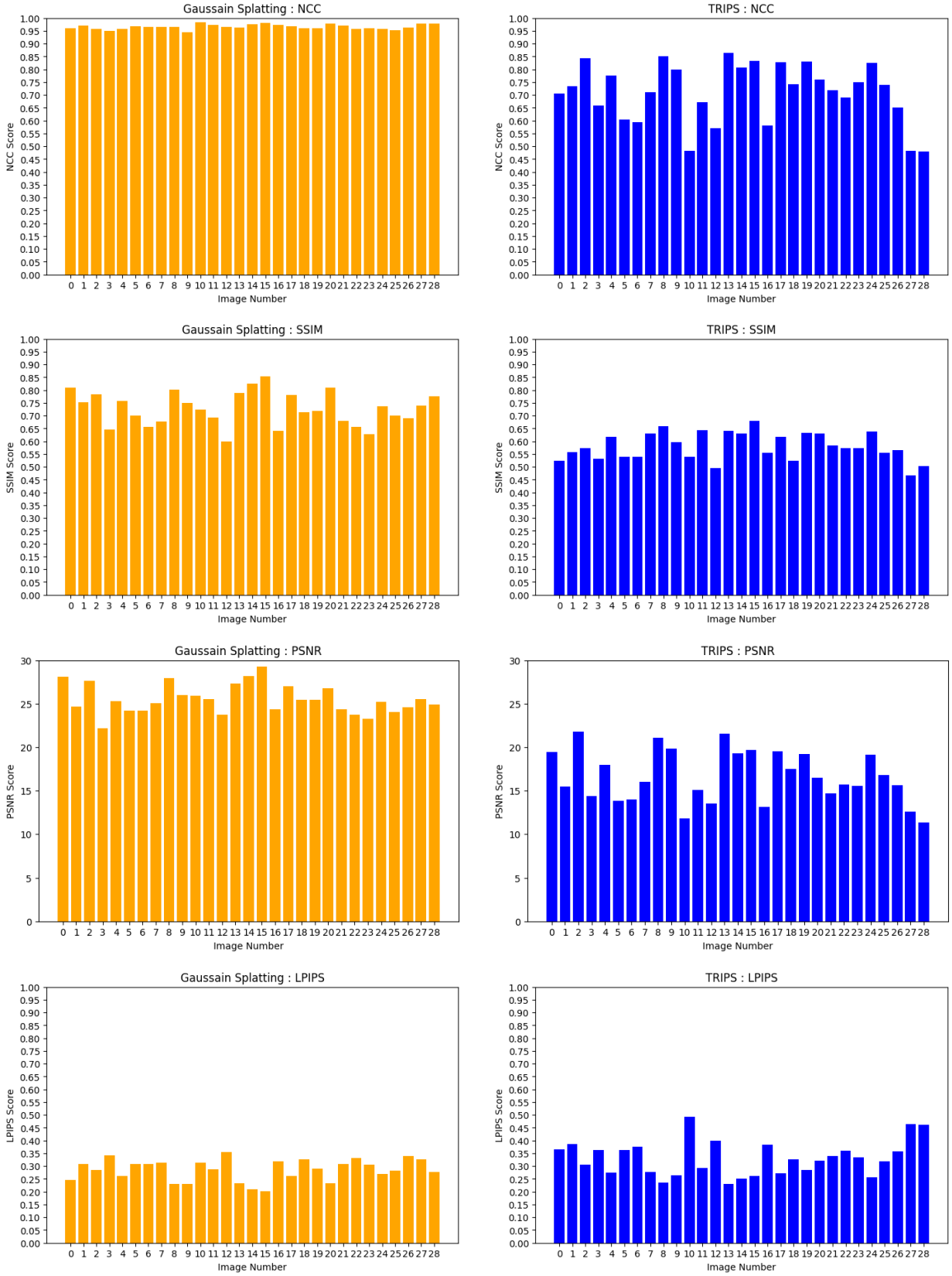


Table 5.2: Breakdown of all metric results for GS and TRIPS

## 5.2 Qualitative Results

When interacting with the results of both solutions in real-time, TRIPS appears to be superior to GS for the application of rock wall climbing. GS struggles with close-up views, with the render quickly devolving when the camera is placed close to the rock wall. As well as this, the result lacks the geometry information needed to appreciate potential holds on the rock face. This is due to the use of a sparse point cloud for GS, which lacks detailed geometry information. This, with the combination of blurry Gaussians up-close make the solution completely unsuitable for rock wall climbing. In comparison, TRIPS uses a dense point cloud, with dedicated optimisation of the wall texture with gated convolution layers. This results in very high fidelity even at extremely close camera positions. It also is more clear due to pixel splatting, and captures the geometry more clearly than GS. This is mainly due to the screen-space pixel-based approach, rather than the world-space based approach of GS. Although GS reaches a high level of quality when training for the input images, due to its compact representation and fast alpha-blending renderer, when rendering different viewpoints, especially at closer distances, it uses the same set of Gaussians. This manifests as a "screen door effect", i.e. as you look at closer distances, the image begins to break down, and you can see the individual Gaussians and gaps in the reconstruction. GS also does not aim to retain the 3-D geometry of the scene, only to render high quality viewpoints from a similar distance to the original images. In comparison to the novel view synthesis methods, the traditional 3-D mesh was far superior at retaining geometry information. Although its texture information was far worse than GS and TRIPS, it would maintain geometric features more clearly. As was mentioned in the Introduction, Poisson meshing smooths over smaller geometries on the rock face, making smaller holds difficult to identify.

Runtime performance of the mesh was poor, reaching only 2-3 fps in MeshLab, however the mesh was not cleaned up in any way and it is likely that the performance could be greatly optimized, however this was outside of the scope of

Figure 5.1: Example of "screen door" effect of GS where individual Gaussians are visible, TRIPS render on left

this project. GS had the best performance, reaching over 100 fps at 1080p resolution, with lows of around 60 fps in detailed areas and higher resolutions, TRIPS was less performant, reaching around 20-30 fps on average with lows under 20 fps. This is due to the more expensive rendering technique used as opposed to GS, which had a highly optimized tile-based rasterizer. TRIPS was also rendering at high resolution (2000x3000), and if the resolution was reduce there could be a gain made in its runtime performance. All methods were run on an NVIDIA RTX A4000 GPU.

## 5.3 Discussion

Although TRIPS does a good job at capturing the close up texture and geometry, while maintaining real-time render performance, it is unclear whether novel-view synthesis methods are suitable for rock wall climbing. Novel view synthesis methods aim to render high quality renders using the input images and their camera positions. This is done in using SfM this case. However, it is difficult to obtain camera positions from very close viewpoints of the rock wall using SfM, as it would it need many more images which would inflate computational cost considerably. SfM finds resolving points from images at a varying distances difficult (11), which could result in a worse point cloud. This means that the images captured for GS and TRIPS are too far away from the subject for the climber to see the smaller holds on the rock face, making it difficult for GS and TRIPS to render these features. As a result, they have to approximate these features, which could result in not only loss of the

features, but incorrect features being added to the render.

In addition to this, they provide no extra detail to the geometry itself, only textural detail. This makes them less versatile to use for different methods that may help climbers, for example, a point cloud can be used for route plotting as is explored by Kajastila and Hämäläinen (2), or for projecting other images onto the point cloud as is done by Whiting et al. (16). When looking at climbing visualisation, it is more important for climbers to mentally visualise and go through the motions of the climb, rather than just identifying the holds. Kramer (1) highlights this by describing how climbers were unable to complete outdoor routes with the same difficulty rating as indoor routes that they were able to complete. Although a 3-D visualisation could be useful for identifying and labelling holds on the rock face, especially holds which are higher up and thus more difficult to see, it can be done using a more traditional methods like SfM with Poisson meshing, which results in a clearer model that can be easily run on consumer machines. In comparison, novel view synthesis methods run on custom graphics pipelines using CUDA kernels, which makes them difficult to run on consumer machines, such as mobile phones.

Climbers also do not plan routes in their entirety before the climb. Rucinska (17) finds that there is an "enactive planning" to climbing, where although there is certain preparation before the climb, the climber still needs to "re-visualise" the climb during the execution of a route. It follows from this that tools that could help climbers "re-visualise" *during* a climb would be far more beneficial than visualisation before the climb.

Illustrative rendering may provide techniques that would be able to capture geometric data in a clearer way that would provide more insight to climbers. For example, the view-dependent feature lines on the holds of the rock face could be extracted and highlighted, which could be used as a learning tool for novice outdoor climbers.

The other image datasets were not used in the comparison of the techniques, mainly

due to practical constraints, however many of the other image datasets produces point clouds and meshes that had gaps in the final reconstruction, which would have made it difficult to compare the results of the rendering techniques, and it was therefore decided to drop the image datasets of the other walls and devices. There are many studies that look into the affect of different devices on the reconstruction process as is outlined by Smith et al. (11).

# 6 | Conclusions and Future Work

This report has investigated the use of 3-D reconstruction and real-time novel view synthesis techniques in climbing environments of outdoor rock walls and compared two state-of-the-art techniques (Gaussian Splatting, and Trilinear Point Splatting) on a set of images of a outdoor climbing wall. It has used both quantitative and qualitative metrics to investigate their quality against traditional reconstruction methods as well as against each other. The study used traditional image-based metrics for quantitative comparison. This included pixel-based metrics like PSNR, NCC, and SSIM, as well as perceptual metrics such as LPIPS. The metrics were run on the output renders of GS and TRIPS. GS was found to perform better when reproducing the input images and had higher metric scores than TRIPS. GS generally had better performance in far-away views than TRIPS. TRIPS struggled with filling the large, textureless regions of the input images, and was also trained with smaller batch sizes, resulting in poorer training performance and render results. In the qualitative comparison between all 3 techniques (GS, TRIPS, and SfM/MVS), it was found that GS performed the worse in close up views, as the Gaussians representing the reconstruction would break down at viewing distances close to the model. TRIPS and SfM/MVS did not have this issue, with both methods retaining the geometry of the model. TRIPS was far superior at retaining high quality texture information up-close, whereas SfM/MVS had blurry textures at close distances. SfM/MVS had the best runtime performance, with GS being worse, but acceptable at 1080p, with TRIPS performing the worse, but still reaching real-time performance.

In the discussion, it was concluded that although novel view synthesis techniques could render views with more detail than traditional methods, their use for rock wall climbing is limited, in part due to their high computational cost, as well as the lack of importance of visual information for learning outdoor climbing routes, or training for difficult outdoor routes in a professional setting. Traditional methods were found to have more merit for climbing rock walls, as additional information can be obtained from point cloud data that can be used for more useful tools (as explored by Whiting et al. (16)). Run-time performance of traditional meshing was far superior to novel view synthesis methods. GS was highly performant at 1080p with 80fps on a NVIDIA RTX A4000. TRIPS performed the worst with a runtime performance of 20-30 fps. Illustrative rendering techniques were not investigated in this work and further research should look into rendering techniques that highlight both model geometry *and* model texture. Further work should be done on capturing images and camera parameters that are close to the rock face, as novel view synthesis would be able to optimize for small features on the subject directly.

Novel view synthesis had developed rapidly and is now able to create high-fidelity reconstructions at interactive frame rates. However, in the context of rock wall climbing, it does not enhance geometric features, and does not help with the task of finding the motions needed to execute a route on a climbing wall.

# Bibliography

[1] Lucas M. Kramer. Rock climbers' perception of movement skill transfer from indoor climbing walls to outdoor natural rock faces. Master's thesis, University of Minnesota, 2023. URL https://hdl.handle.net/11299/256448.

[2] Raine A. Kajastila and Perttu Hämäläinen. Benefits of 3d topos for information sharing and planning in rock climbing. *Sports Technology*, 7:128 – 140, 2014. URL https://api.semanticscholar.org/CorpusID:109046498.

[3] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *CoRR*, abs/2003.08934, 2020. URL https://arxiv.org/abs/2003.08934.

[4] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5855–5864, October 2021.

[5] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5470–5479, June 2022.

[6] Linus Franke, Darius Rückert, Laura Fink, and Marc Stamminger. Trips:

Trilinear point splatting for real-time radiance field rendering, 2024. URL https://arxiv.org/abs/2401.06003.

[7] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023. URL https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/.

[8] Richard A. Robb. 3-d visualization in biomedical applications. *Annual Review of Biomedical Engineering*, 1(Volume 1, 1999):377–399, 1999. ISSN 1545-4274. doi: https://doi.org/10.1146/annurev.bioeng.1.1.377. URL https://www.annualreviews.org/content/journals/10.1146/annurev.bioeng.1.1.377.

[9] M. Scaioni, J. Crippa, M. Corti, L. Barazzetti, D. Fugazza, R. Azzoni, M. Cernuschi, and G. A. Diolaiuti. Technical aspects related to the application of sfm photogrammetry in high mountain. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2: 1029–1036, 2018. doi: 10.5194/isprs-archives-XLII-2-1029-2018. URL https://isprs-archives.copernicus.org/articles/XLII-2/1029/2018/.

[10] M. Scaioni, M. Corti156, G. Diolaiuti, D. Fugazza, and M. Cernuschi. Local and general monitoring of forni glacier (italian alps) using multi-platform structure-from-motion photogrammetry. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W7: 1547–1554, 2017. doi: 10.5194/isprs-archives-XLII-2-W7-1547-2017. URL https://isprs-archives.copernicus.org/articles/XLII-2-W7/1547/2017/.

[11] M.W. Smith, J.L. Carrivick, and D.J. Quincey. Structure from motion photogrammetry in physical geography. *Progress in Physical Geography: Earth and Environment*, 40(2):247–275, 2016. doi: 10.1177/0309133315615805. URL https://doi.org/10.1177/0309133315615805.

[12] Leandro Bornaz, Fulvio Rinaudo, et al. Terrestrial laser scanner data processing. In *XXth ISPRS Congress Istanbul*, 2004.

[13] A. Rocha, R. C. R. Mota, H. Hamdi, U. R. Alim, and M. Costa Sousa. Illustrative multivariate visualization for geological modelling. *Computer Graphics Forum*, 37 (3):465–477, 2018. doi: https://doi.org/10.1111/cgf.13434. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13434.

[14] Kai Lawonn, Ivan Viola, Bernhard Preim, and Tobias Isenberg. A survey of surface-based illustrative rendering for visualization. *Computer Graphics Forum*, 37(6):205–234, 2018. doi: https://doi.org/10.1111/cgf.13322. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13322.

[15] S. Ruess, G. Paulus, and K.-H. Anders. The use of high-resolution photogrammetry for the survey and analysis of rock-climbing walls. *AGILE: GIScience Series*, 3:58, 2022. doi: 10.5194/agile-giss-3-58-2022. URL https://agile-giss.copernicus.org/articles/3/58/2022/.

[16] Emily Whiting, Nada Ouf, Liane Makatura, Christos Mousas, Zhenyu Shu, and Ladislav Kavan. Environment-scale fabrication: Replicating outdoor climbing experiences. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 1794–1804, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450346559. doi: 10.1145/3025453.3025465. URL https://doi.org/10.1145/3025453.3025465.

[17] Zuzanna Rucińska. Enactive planning in rock climbing: recalibration, visualization and nested affordances. *Synthese*, 199(1):5285–5310, December 2021.

[18] M. L. Cappuccio. *Handbook of embodied cognition and sport psychology.* The MIT Press., 2019.

[19] Aditya M. Intwala and Atul Magikar. A review on process of 3d model reconstruction. In *2016 International Conference on Electrical, Electronics, and*

*Optimization Techniques (ICEEOT)*, pages 2851–2855, 2016. doi: 10.1109/ICEEOT.2016.7755218.

[20] Takashi Oguchi, S. Hayakawa Yuichi, and Thad Wasklewicz. Chapter seven - data sources. In Mike J. Smith, Paolo Paron, and James S. Griffiths, editors, *Geomorphological Mapping*, volume 15 of *Developments in Earth Surface Processes*, pages 189–224. Elsevier, 2011. doi: https://doi.org/10.1016/B978-0-444-53446-0.00007-0. URL `https://www.sciencedirect.com/science/article/pii/B9780444534460000070`.

[21] Zhen Dong, Fuxun Liang, Bisheng Yang, Yusheng Xu, Yufu Zang, Jianping Li, Yuan Wang, Wenxia Dai, Hongchao Fan, Juha Hyyppä, and Uwe Stilla. Registration of large-scale terrestrial laser scanner point clouds: A review and benchmark. *ISPRS Journal of Photogrammetry and Remote Sensing*, 163:327–342, 2020. ISSN 0924-2716. doi: https://doi.org/10.1016/j.isprsjprs.2020.03.013. URL `https://www.sciencedirect.com/science/article/pii/S0924271620300836`.

[22] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[23] Richard I. Hartley and Peter Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2):146–157, 1997. ISSN 1077-3142. doi: https://doi.org/10.1006/cviu.1997.0547. URL `https://www.sciencedirect.com/science/article/pii/S1077314297905476`.

[24] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[25] Stéphanie Aravecchia, Marianne Clausel, and Cedric Pradalier. Comparing metrics for evaluating 3d reconstruction quality in large-scale environment mapping. In *in: Computer Vision–ECCV 2023: 18th European Conference, Budapest,*

*Hungary, September 18-23, 2023, Proceedings, Part IV*, pages 300–315. Springer, 2023.

[26] Z. Wang, E.P. Simoncelli, and A.C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems  Computers, 2003*, volume 2, pages 1398–1402 Vol.2, 2003. doi: 10.1109/ACSSC.2003.1292216.

[27] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[28] H.R. Sheikh and A.C. Bovik. Image information and visual quality. *IEEE Transactions on Image Processing*, 15(2):430–444, 2006. doi: 10.1109/TIP.2005.859378.

[29] Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks, 2021. URL `https://arxiv.org/abs/2112.05131`.

[30] Rosalie A Waelen. The ethics of computer vision: an overview in terms of power. *AI and Ethics*, 4(2):353–362, 2024.

[31] Kate Crawford and Trevor Paglen. Excavating ai: the politics of images in machine learning training sets. *AI & SOCIETY*, 36(4):1105–1116, Dec 2021. ISSN 1435-5655. doi: 10.1007/s00146-021-01162-8. URL `https://doi.org/10.1007/s00146-021-01162-8`.

[32] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, page 61–70, Goslar, DEU, 2006. Eurographics Association. ISBN 3905673363.

[33] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29(4):551–559, 1983. doi: 10.1109/TIT.1983.1056714.

[34] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, 1999. doi: 10.1109/2945.817351.

[35] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Trans. Graph.*, 37(6), December 2018. ISSN 0730-0301. doi: 10.1145/3272127.3275084. URL `https://doi.org/10.1145/3272127.3275084`.

[36] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S. Huang. Free-form image inpainting with gated convolution. *CoRR*, abs/1806.03589, 2018. URL `http://arxiv.org/abs/1806.03589`.

[37] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution, 2016. URL `https://arxiv.org/abs/1603.08155`.

[38] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In Vittorio Scarano, Rosario De Chiara, and Ugo Erra, editors, *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008. ISBN 978-3-905673-68-5. doi: 10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136.