Examination of Minimal Camera Setups for Volumetric Video Capturing from a Time Perspective

Marek Magala[1], Ivan Minárik[1]
[1] Faculty of Electrical Engineering and Information Technology,
Slovak University of Technology in Bratislava, Ilkovičova 3, 84104 Bratislava
xmagala@stuba.sk

Abstract – **This study examines the impact of camera count on training time and quality in volumetric video capture using 3D Gaussian Splatting. Two datasets were processed with varying camera numbers, testing two reduction strategies: removing cameras from the center or edges. Results show that 14 cameras offer an optimal balance, reducing training time by up to 32 hours per minute of video (30 fps) with minimal quality loss, while fewer cameras cause significant artifacts. Edge removal proved more efficient, minimizing computational errors.**

Keywords – **Gaussian Splatting; Volumetric video; Time of training; Quality.**

I. Introduction

Volumetric video captures dynamic 3D scenes, enabling applications in VR, AR, CGI, and videotelephony. However, high computational costs and long training times hinder real-time use. This project aims to identify the minimal camera setup that balances training efficiency and model quality, using 3D Gaussian Splatting to process video frames from two datasets. The main goal of this project is to reduce long training times without sacrificing too much of the quality of the models. Finding this balance is key if volumetric video is going to become a more practical and widely used tool. Throughout the project, we tested different methods for removing cameras to optimize the training process. Although the original plan was to use 4D Gaussian Splatting, technical limitations led us to adapt differrent approach and instead use standard Gaussian Splatting, training a separate model for each frame of video.

Workstation, where research was conducted used processor AMD Ryzen 9 7950X with 16 cores, 64 GB of RAM memory and graphic card Nvidia GeForce RTX 3060 Ti.

II. Related work

Volumetric video technology has made significant advances in recent years, driven by improvements in camera systems, computer vision algorithms and neural rendering techniques. One promising recent development is Gaussian Splatting, which represents scenes as collection of Gaussian functions rather than traditional meshes. 3D Gaussians are geometrical object shaped as ellipsoid, which are defined by position, opacity α, color and covariance matrix Σ which defines stretch and scaling of the Gaussian. The covariance matrix Σ is model with scaling matrix S and rotational matrix R.

$$\Sigma = RSS^T R^T \tag{1}$$

Definition of 3D Gaussian position for specific point with center in point μ inside of 3D field is computed with equation:

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

This representation allows effectively model 3D structures, although this process can by optimized. Projecting of the scene runs by iterative rendering and matching 3D scene with real life views from images of dataset. From every image it will render view on scene and then every view is compared with real life views. Because of this process the function of loss L is minimized. Loss function in for interpreting difference between real life scene from generated one.

$$L = (1 - \lambda)L_1 + \lambda L_{D-SSIM} \tag{3}$$

L_1 indicates difference in pixels between predicted and real image while $L_{D\text{-}SSIM}$ indicates structural differences while λ determines the percentage share that $L_{D\text{-}SSIM}$ has in the total loss function L and is experimentally set to $\lambda = 0.1$ [1].

4D Gaussian Splatting extends this idea 3D Gaussians of time domain. The technique we originally intended to use [2] defines canonical 3D Gaussians. These canonical Gaussians are modeled from the first frame and then moving is modeled. In this method spatial-temporal structure encoder is used. This encoder groups all nearby Gaussians by which computing of moving is more precise. It also uses a small neuron network which is used for decoding of moving and deformation. The final projection is used by differential splatting and for optimization total-variational loss is used to reduce big changes between two frames in time.

Gaussians before there modeling needs cloud of points. This point cloud is made from a process called Structure from Motion (SfM). This process predicts 3D structure from collection of 2D images. One of the core methods involves identifying corresponding points across images taken from different viewpoints. If a specific, significant point is captured by two or more cameras, it can be identified using epipolar geometry.

Significant points in the images can be detected using various algorithms, such as the Harris corner detector or SIFT. Big part of the process is determining the camera parameters, including calibration and position, which are represented by a projection matrix. This step involves a process known as bundle adjustment, which minimizes the reprojection error.

Reprojection error is defined as a difference that occurs when calculated 3D points are projected back onto their original 2D images. By reducing these errors, the accuracy of the reconstructed 3D scene improves, and inconsistent 3D points are eliminated [3].

Then after SfM Multi-View Stereo (MVS) builds upon its results and serves as a complementary technique. The sparse set of points generated by SfM is used as a foundation for creating a denser set of points, enabling detailed reconstruction of the scene. MVS calculates depth and normal information for each pixel in the images. By combining the depth and normal maps from multiple views, a dense point cloud is produced, providing a detailed representation of the scene. From this dense point cloud, a continuous 3D surface of the scene can be estimated using surface reconstruction algorithms, such as Poisson surface reconstruction. In this work, the open-source software COLMAP [4] was used to perform both the SfM and MVS processes.

COLMAP is general purpose research software tool designed for Structure from Motion and Multi-View Stereo, offering command line or graphical interface. It enables the reconstruction of 3D models from both ordered and unordered sets of images. To use COLMAP, only the path to the folder containing the input images is required; the software then automatically generates a dense point cloud. COLMAP this information stores into 3 .txt files called Cameras, Images and Points3D.

Recent study [5] found a new approach to initial settings. Result is 3DGS-MCMC approach, to make training more robust to initialization via Markov Chain Monte Carlo sampling. This type of sampling gives high probability to collection of Gaussians which are more faithful to real life images.

III. Method

Two datasets, "coffee_martini" and "cut_roasted_beef," were processed by converting videos to images at 2 fps. COLMAP performed Structure-from-Motion (SfM) to generate point clouds, followed by 3D Gaussian Splatting for per-frame model training. Since this process was time consuming, all commands were pre-written into a text file and copied into a terminal. All frames convert sequentially after the last one were converted which ensured smooth and continuous processing of each frame.

After conversion, the *train.py* script was launched to start the actual model training. Script *train.py* is part of Gaussian Splatting workflow from [1]. The final step of training involved copying all frames .ply into one separate folder. In this folder, all trained models are organized and renamed according to the corresponding frame sequence number. At the and all times were measured and written down into spreadsheet. For the second dataset, loss information was also written down.

For removing camera views from dataset were used 2 different methods.

1) Method: Removing cameras positioned in the middle between two cameras along the plane. This approach led to many problems in calculating camera positions, especially when the number of cameras was low (e.g., 6 cameras). This resulted in increased training time for problematic frames up to three times

2) Method: Removing cameras from the outer edges of the captured scene. This method eliminated issues with calculating camera positions. However, it reduced the overall capture area, which negatively affected the quality of the resulting models.

Number of cameras used	Coffee martini [m, sec]	Coffee martini_2 [m, sec]	Cut roasted beef [m, sec]
20	-	-	653,59
18	738,35	738,35	607,05
14	734,28	702,39	601,52
10	621,46	645,3	530,34
6	735,51	506,46	3130,57

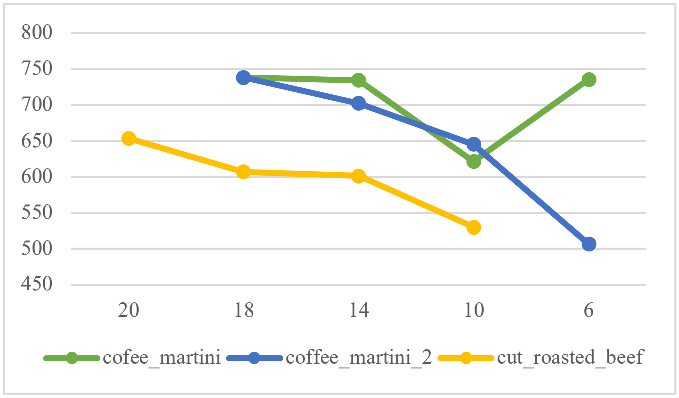

Figure 1. Dependence of training time on the number of cameras per frame

From the Figure 1. is clear to see that the second method was more efficient for almost every number of cameras. The only exception was at 10 cameras, where the first method proved slightly more efficient. Although a difference of 24 minutes may seem significant, the average difference in training time per frame between the two methods for 10 cameras is approximately 1 minute and 12 seconds. Dataset *cut_roasted_beef* was removed from the Figure 1. Because of better visibility of the smaller differences between other datasets.

On the other hand, a substantial difference is noticeable at six cameras, where the total training time difference is nearly 3 hours and 50 minutes. For each frame this corresponds to an average difference of about 11 minutes and 30 seconds. It is important to note that this significant difference does not mean that every model took 11,5 minutes longer to train. Instead, problematic frames sometimes required up to 86 minutes of training more, while “normal” non-problematic frames typically took around 25 minutes.

On the other hand, a substantial difference is noticeable at six cameras, where the total training time difference is nearly 3 hours and 50 minutes. For each frame this corresponds to an average difference of about 11 minutes and 30 seconds. It is important to note that this significant difference does not mean that every model took 11,5 minutes longer to train. Instead,

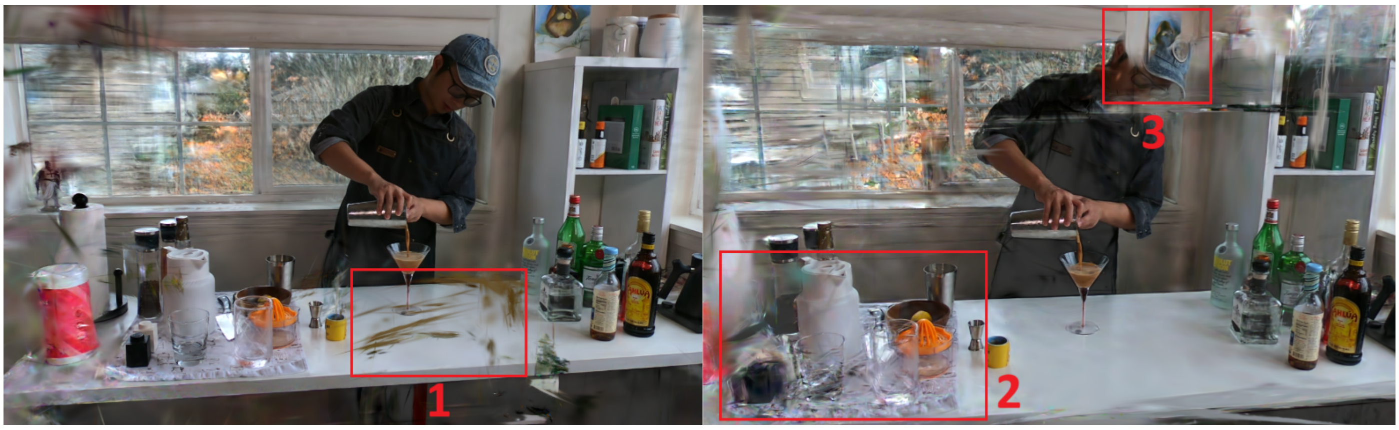

Figure 2. Visible differences between using 18 and 6 cameras in dataset

problematic frames sometimes required up to 86 minutes of training more, while "normal" non-problematic frames typically took around 25 minutes.

IV. Comparison of Models Based on the Number of Cameras

Each model was trained at 30 000 iterations and was named using the format *coffee_martini_00 or cut_roasted_beef*, where the last two digits indicated the total number of cameras used. Additionally, models labeled *coffee_martini_00_2* were created, where the *_2* indicated that camera removal followed the second method. Dataset *cut_raosted_beef* was trained only with second method, because this dataset was used after discovering second method. Difference in dataset which made significantly different results was, that *cut_roasted_beef* was darker dataset with less light in scene.

Since the models were trained using standard Gaussian Splatting, there were occasional issues with estimating the correct position and spread of individual Gaussians. This sometimes resulted in clouds of Gaussians appearing in the final model, which could obstruct the view. These clouds are marked in 0as rectangle 1. and instance where this cloud obstructs the view is visible in Figure 3 in rectangle 1. In extreme cases, parts of the original input photographs could even be seen, where the model failed to properly infer the Gaussian's position. Instance of this can be see in Figure 2. inside of rectangle 3. Objects placed on the table, while distinguishable from one another, were reconstructed with enough quality to easily tell them apart. The window forming the background is not captured in full detail, but its structure and the objects behind it are still recognizable.

The model with the highest number of cameras is *coffee_martini_18*, utilizing 18 cameras. This model required the longest training time but also produced the highest quality result. It offers the widest range of motion, allowing the object to be observed with fewer artifacts. *Cut_roasted_beef* highest number of cameras was 20. The difference in quality of the 20 cameras dataset and 18 cameras dataset was nearly identical. Because of the darker scene in *cut_roasted_beef* the difference with *coffee_martini* is more noticeable. Bigger gaussian clouds, more distortion on the edges.

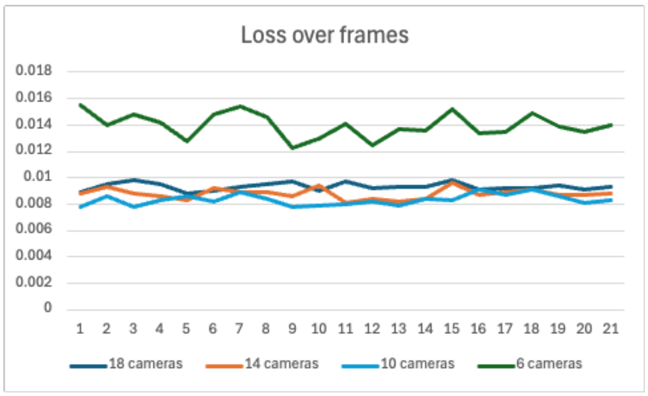

Figure 3. Graph shows development of loss over frames of the video. As expected, lowest camera count renders highest amount of loss. Conversely, loss is relatively stable regardless of whether 10, 14 or 18 cameras are used for reconstruction, with 18 cameras' loss being slightly on top.

The models were not otherwise modified. The only adjusttment made were to align the default camera view directly toward the man in the video. This adjustment ensured that during playback of the volumetric video, the camera remained approximately focused on the same location across different frames. This alignment was done using Jawset Postshot software. The only adjusted parameter was the rotation angle along the Y-axis, which varied between 5° and 40°.

A. 14 Cameras

The first model with removed cameras was *coffee_martini_14*. This model required about 4 minutes less training time compared to the full model. However, the small time difference was mainly due to frame 17, which turned out to be problematic. Frame 17 failed to calculate camera positions and point placements, resulting in a model consisting only of a flat image with no Gaussians produced.

For the other frames, there were only minimal differences compared to models trained with the full number of cameras. However, the artifacts visible in 0present in *coffee_martini_18* became more pronounced, Gaussian clouds were larger and more frequent. Some objects on the edge appeared with lower quality, and the area around the center showed less distortion compared to the edges.

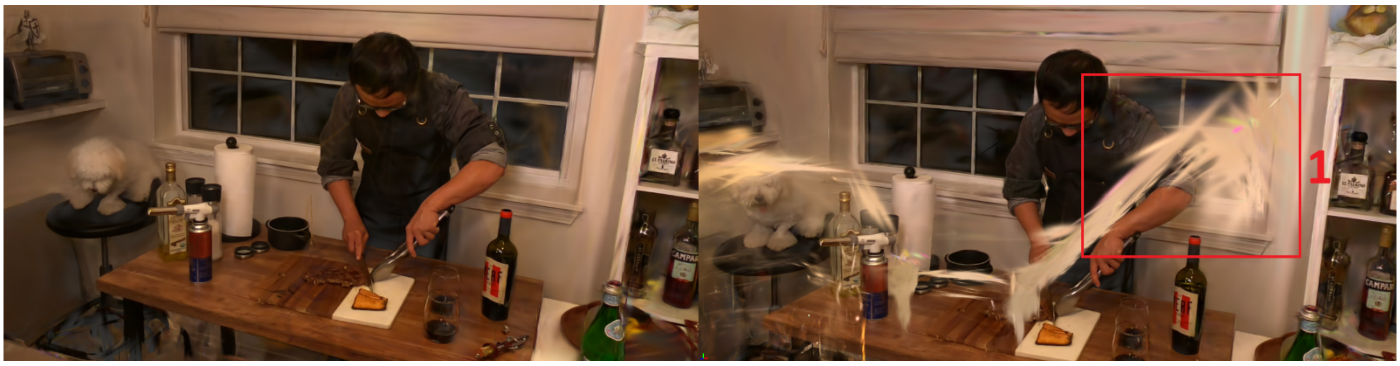

Figure 4. Difference between 20 and 10 cameras in dataset cut_roasted_beef

The *coffee_martini_14_2* model, created using the second removal method, achieved similar results, with the notable improvement that frame 17 was "fixed," leading to an overall shorter volumetric video training time. The main difference in quality between the first and second methods was that, in the second method, objects on the edge visible in 0had more dispersed Gaussians, and quality degraded more quickly when the camera moved away from the center.

Cut_roasted_beef_14 required about 5 minutes less than with 18 cameras, but also the quality was nearly identical. Same Gaussian clouds, items on the edge unexpectedly had in some frames less distortion than with 18 cameras.

B. 10 Cameras

Coffee_martini_10 and *coffee_martini_10_2* already sho-wed more significant differences compared to the full model. Gaussian clouds became more frequent toward the edges of the model, which somewhat obstructed the view, although the freedom of camera movement remained similar. On the edges, the Gaussian artifacts could block parts of the model. In the center, the man's lower torso and legs, which should have been black, appeared gray instead.

Cut_roasted_beef_10 continued trend of taking less time for training, without sacrificing too much of quality in model. The difference between models with 18 cameras and 10 had almost no visible differences apart from the edges, where distortion of gaussians was earlier than with more cameras.

C. 6 Cameras

Coffee_martini_6 was the model with the fewest cameras, leading to significant degradation. In 7 out of 20 frames, the scene was almost unrecognizable. Most often, only a flat image without proper 3D structure was created. Even the frames that were successfully modelled contained large amounts of Gaussian noise. This model also took longer to train than model with all cameras.

Finally, *coffee_martini_6_2*, despite not containing problematic frames, remained largely unusable. The movement away from the model's center was too restricted for meaningful volumetric video playback. Gaussian clouds were very frequent and large. The man's black coat turned gray, and in some frames, his head was visibly displaced from the body.

Cut_roasted_beef_6 was so ineffective model, that the time of training was removed from Figure 1. While quality difference between 10 cameras and 6 were almost unrecognizable.

V. Results

Training time decreased with fewer cameras, with edge removal being more efficient (e.g., 31% reduction at 6 cameras). However, quality degraded significantly below 10 cameras, with artifacts and failed 3D reconstructions. The 14-camera setup was optimal, saving around 32 hours for a 1-minute video (30 fps) compared to 18 cameras, with negligible quality loss. Darker datasets showed less sensitivity to camera reduction.

Although the 10-camera setup saved some training time, this reduction significantly impacted model quality, which noticeably declined. With the 6-camera setup, time savings were inconsistent. In some cases, training took up to six times longer compared to using all cameras, while model quality deteriorated.

Method 2 of camera removal was more efficient, reducing errors in camera pose and point cloud estimation. However, it couldn't fully offset quality loss with fewer cameras, where models fell below acceptable standards.

Acknowledgement

Research in this paper was supported by projects DISIC (09I05-03-V02-00077), InteRViR (VEGA 1/0605/23), and NEXT (ERASMUS-EDU-2023-CBHE-STRAND-2, ID: 101129022).

References

[1] KERBL, B.; KOPANAS, G.; LEIMKÜHLER, T.; DRETTAKIS, G. 3D Gaussian Splatting for Real-Time Radiance Field Rendering [online]. ArXiv. 2023. [cit. 2025-04-20] Available from: https://arxiv.org/pdf/2308.04079

[2] WU, G.; YI, G.; FANG, J.; XIE, L.; ZHANG, X.; WEI, W; LIU, W. TIAN, Q.; WANG, X. 4D Gaussian Splatting for Real-Time Dynamic Scene Rendering [online]. *ArXiv.* 2024-7-15. [cit. 2025-04-20]. Available from https://arxiv.org/pdf/2310.08528v3.

[3] LOBO, T. Understanding Structure From Motion Algorithms [online]. *Medium.* 2023-12-18. [cit. 2025-04-20]. Available from https://medium.com/@loboateresa/understanding-structure-from-motion-algorithms-fc034875fd0c.

[4] SCHOENBERGER, J. COLMAP [online]. *GitHub.* 2024. [cit. 2025-04-22]. Available from https://colmap.github.io/tutorial.html.

[5] HUANG, Z.; WANG, P.; ZHANG, J.; LIU, Y.; LI, X.; WANG, W. 3R-GS: Best Practice in Optimizing Camera Poses Along with 3DGS [online]. ArXiv. 2025-04-05. [cit. 2025-04-21]. Available from: https://arxiv.org/pdf/2504.04294v1.pdf

[6] T. Li *et al.*, "Neural 3D Video Synthesis from Multi-view Video", v *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, New Orleans, LA, USA: IEEE, jún. 2022, s. 5511–5521. doi: 10.1109/CVPR52688.2022.00544.