# Virtual Tour Using Gaussian Splatting

Samuel Lančarič, Marek Vančo
Faculty of Electrical Engineering and Information Technology, Slovak University of Technology Bratislava, Slovakia
xlancarics@stuba.sk, marek_vanco@stuba.sk

*Abstract* - **This bachelor thesis includes the creation of an immersive tour and the creation of the most accurate copy of the school as a 3D model in which the user will be able to move freely, interact with the objects and especially to allow foreign students to visit the corridors of the faculty FEI STU in this way. To create the 3D model, I will use the technology "Gaussian splatting", which is designed to create 3D representations of objects and spaces from 2D data to 3D objects. Thanks to this technology we can create one or more 3D models from a number of images. I will use the program "Jawset, Postshot"[1] to process the data consisting of images of the school premises. I will also use the game development program "Unreal engine" in my research, which has direct compatibility with the 3D models created in "Jawset, Postshot"[2]. This compatibility will make the creation and thorough finalization of the project possible and much easier.**

***Keywords*** **- Gaussian splatting; Virtual navigation; 2D-to-3D data processing; Digital twin; object**

## I. Introduction

Throughout the last couple of years, the technology for creating 3D objects has progressed from long-used Photogrammetry[3] to Gaussian splatting. Both technologies use a lot of 2D data in the form of photographs to create a 3D object. It's incredible how these technologies are constantly improving and taking the possibilities of creating 3D objects to a new level.

Photogrammetry is the main technology for creating objects from the images of a given object. The first applications for creating 3D objects were in the 1980[4] Thanks to the data provided, the technology creates a point cloud that represents the surface points of the object, using these points we can construct a mesh that represents the form and structure of the object. Disadvantage of creating model like this, is that the quality of the object is only as good as the mash is. Quality of the mash depends on the number of points in the point cloud. When creating mash from a data set of a room, blank walls are usually with little to none points in the point cloud. The same happen when we try to create mash from reflective surfaces or windows. This creates holes in the mash and do not create the textures there. The last step is the application of colored textures on the surface.

Gaussian splatting, unlike photogrammetry, needs a point cloud from which it does not create a mesh, but uses the provided data to add Gaussians which are represented as ellipsoidal objects with their color and opacity. Then placing these "clouds" into the 3D approach creates our object. Even when there are no points in the point cloud, the gaussian splats are being placed on those locations. Gaussian splatting is more targeted on the data set and place the splats according to the photos provided and not mainly from the point cloud. That is why is crucial to have as detailed and clean data set with no blur in the images and even exposure. At this point technology can only create parts of the model only from the taken images. If some part of an object is missing in the data set, it will not be generated into the model.

## II. Use of Gaussian Splatting

The aim of the work was to combine the new technology of gaussian splatting to create a tour of the school. In the past years, this technology has been used to represent t single objects (e.g., a model of a stone or a model of a car). Thanks to the creation of the digital twin, this is a perfect application to the gaming industry. The use of gaussian splatting to represent a room is more demanding due to the large amount of data needed to represent the room correctly, but the quality of the model is several times better than in photogrammetry, however the speed remains the same.

## III. Workflow pipeline

- Data collecting (images/videos)
- Editing images
- Import into the appropriate program (e.g., Jawset Poshshot)
- Correction of errors on the created model
- Import the model into the game engine

### *A. Data collecting*

- An ordinary camera from a phone or even a 360º camera can be used to collect data. The main thing is that the images are clear, without any interference from direct sunlight and that everything can be seen in the image. For the most accurate images we try to move the camera and not tilt it.
- Using the wide-angle mode on phones is not recommended due to not keeping the corners square and distorting the edge of the image. Also, when using video from a 360º camera, it is necessary to create 6 different parts of the image from one frame as a projection on a cube so that the corners are preserved and no parts are distorted.
- The best method for collecting data, is to make 3 circles around the object at different heights, the first at the bottom slightly tilted upwards, the second at chest height oriented horizontally, and the last at overhead height slightly tilted downwards. Next, we need to take

photographs of the objects in the room from different angles for a proper picture.

### B. *Editing images*

When editing photos, we try to adjust the white balance so that every photo has the same tone. Thanks to this, we can achieve a uniform creation of a single-colored wall in the room. Than we edit the other parts of the photo where there is a high exposure of the sun and no dark spots while we try to keep the quality of the image at maximum.

### C. *Import into the appropriate program*

The creation of the model itself consists of several steps that can limit the quality of the model. Jawset postshot provides an end-to-end process for model creation. The steps in creating a model are:

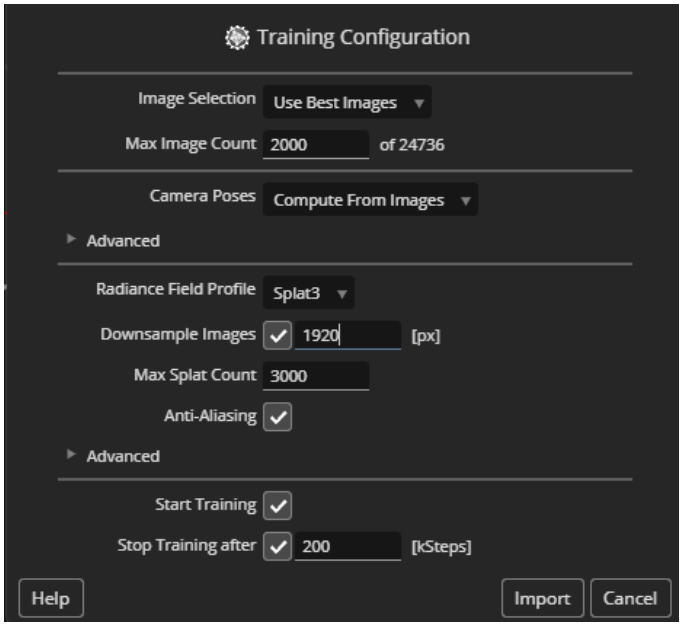


Figure 1. Input setting when importing data as a video

#### a) *Selecting the best images*

In the process of selecting the best images, we first select the images that are blurry or of insufficient quality that we will not use. When using data from the video, the program itself selects the frames and especially the number of frames to create the best quality model or we can use script to select given number of photos manually. The recommended frame selection for video is approximately 4-5 frames for each second of video. Of course, this number is only a guideline.

#### b) *Calculating the camera position for each frame*

To calculate the position of each frame, the Structure-from-Motion algorithm is used, which can determine the camera position for that frame, thanks to the control points on it. It is not convenient that there are not enough points on the frames to give the exact position. For a white wall image where there is no detail, it is difficult, almost impossible for this technology to determine the position of the image. Similarly, images with only ellipsoidal features are not easy to calculate the position. When collecting the data, we try to capture a pair of clues on each image that will allow us to determine the position of the image.

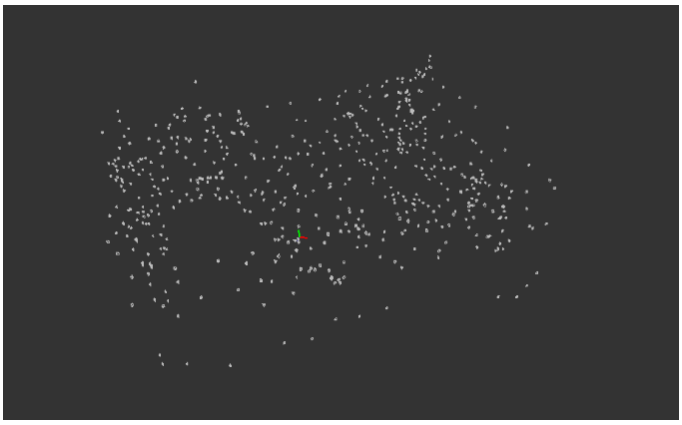

Figure 2. Representation of all aligned images taken in B406 at FEI STU

#### c) *Creating a point cloud*

To create a point cloud, Structure-from-Motion is also used, which tries to find details from the frames it has correctly positioned. Therefore, assigns a point in space to the details and color changes in the frames. With all the points assigned we can usually see the outline of the object we are trying to model. With this point cloud we can tell if the modeling will be of sufficient quality, more points equal more detail in a given location which equals to better quality.

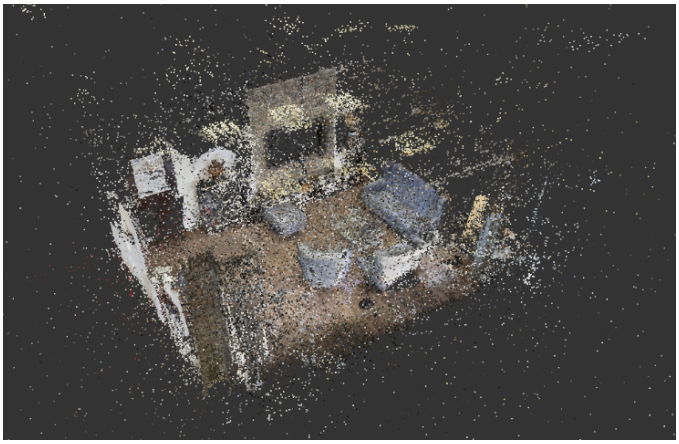

Figure 3. Representation of a point cloud generated from the alignment of a B406 class at FEI STU

#### d) *Assigning Gaussians to each point*

A very large number of gaussians are used to create the model, which can be read as an ellipsoidal object in space. This gaussian has its own position, dimension, rotation, color and opacity. Thanks to the large number of these elements, we can create our model. For larger and more detailed models, we can use up to several million of these elements. The disadvantage of these models is the poor quality in places, where there were not enough frames or poor quality of the frames. In this case, the models will show artifacts or floaters with poor quality.

### D. *Correction of errors on the created model*

After the successful and high-quality creation of the model, it is still necessary to manually edit and clean the created model from such floating artifacts. This option is also capable directly in the "Postshot" program. In addition to errors, floaters can also appear in the model, these artefacts are most often found near the walls of the room. To minimize these artifacts, we can take a couple of shots right next to the wall in the room.

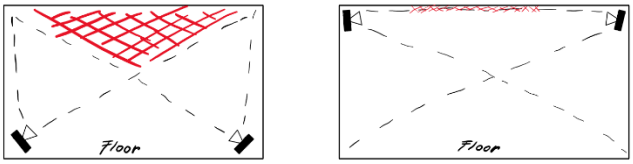


Figure 4. Illustration of the space for floaters to occur if the data is not collected properly

### E. *Import the model into the game engine*

These 3D models can then be inserted into a game development program. The two primary programs for this are "Unity" and "Unreal engine", but it is necessary to implement a plug-in that can support the display of the model. For the “Unity” development environment, the plug-in from the author "Aras Pranckevičius" can be used, who has provided this plug-in as an open-source project on the “GitHub” platform. For the “Unreal engine” development environment, the Jawset postshot plug-in has been released by the Jawset developers, which directly supports .psht and not .ply files. This makes working with models smoother and faster[5].

## IV. DISADVANTAGES OF GAUSSIAN SPLATTING

Because the model is just a graphical representation of the object, the model itself has no mesh, something to define its collision elements in space. By the fact that it is only many ellipsoidal objects, it is necessary to either manually create a collision field and collision parameters for each object, or, thanks to programs that are designed for photogrammetry, to create a collision mesh, which is then synchronized with the model.

### A. *Creating a mesh*

To create the mesh, we use the “Reality capture” application [6], which is designed to create a model using photogrammetry technique. Reality capture is an excellent tool for quick and easy mesh creation. We choose to import the whole data set we have. “Reality capture” prioritize faster computation and will not put every single picture to one component, rather creates multiple smaller components and we can use control points to manually connect the components to single one. For our application it is not required to have perfect component to create mash from. To speed the process, we can only use the biggest component that it generated. We try to keep the mesh to a minimum quality for smoother gameplay.

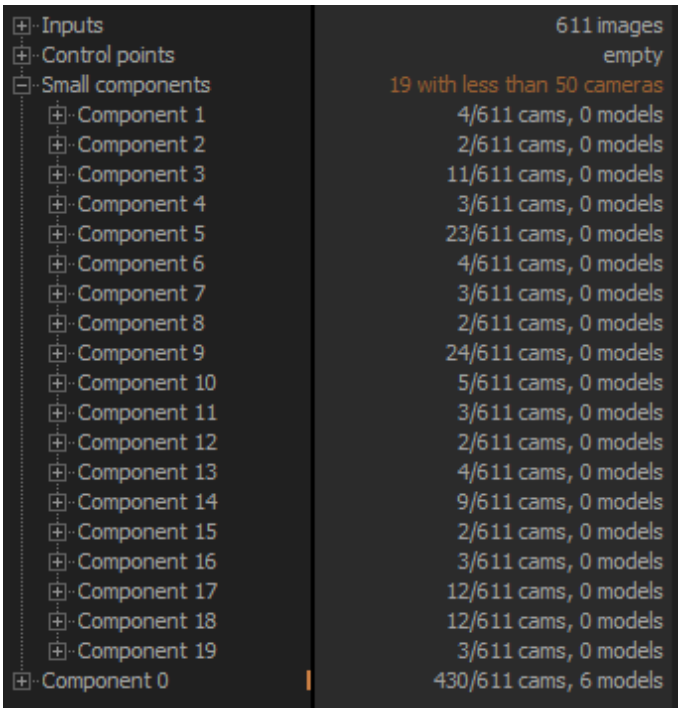


Figure 5. List off all the component generated by Reality Capture from data set containing 611 images of class B406 at FEI STU

### B. *Importing the mesh*

Importing mesh is straightforward. File containing the mash must be in the .fbx format so the Unreal engine can understand that it is a mash that we want to import. Then it is enough to synchronize the position and size of the mesh and the model. Unreal engine offers the option to add the mash as the component to the model itself, this makes resizing and changing the location of the model also resize and change location of the mash according to the model.

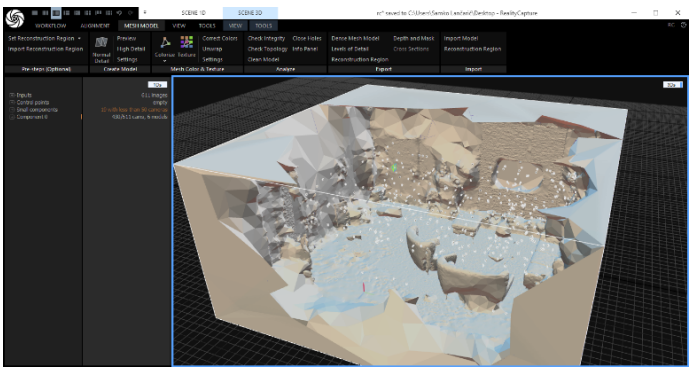

Figure 6. Mesh created in Reality Capture using just the biggest component from aligning images of class B406 at FEI STU

## V. RESULTS

The result of the work is a complete functional application, ensuring immersive experience in first person perspective within the recreated FEI STU building. The most essential advantages and functions of the application are:

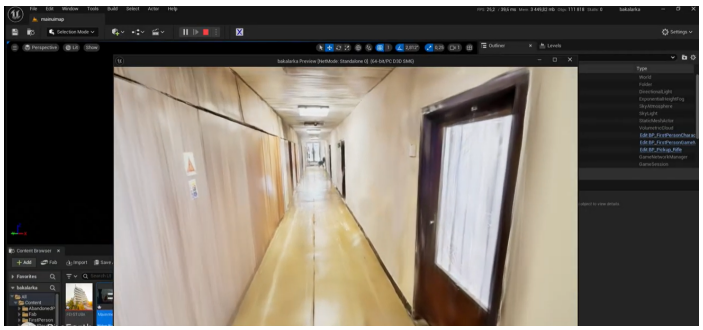

Figure 7. Example of the hallway at FEI STU implemented in Unreal engine

- Support for different hardware configurations.

The environment uses LOD (level-of-detail) techniques [7] and intelligent data streaming, so it is easy to run even on less powerful PCs without compromising any quality.

- Precise collision thanks to mesh hitboxes

Gaussian splats are used to display highly detailed surface graphics, while optimized mesh colliders (hitboxes) are used for collisions. This ensures that player movement and possible interactions are accurate and without unwanted penetration through walls or furniture.

- Modular architecture and scalability

Each room or faculty department is done as a separate sub-scene (level), thereby allowing for easy introduction of new rooms, floors or whole buildings without major modifications to the existing project.

- Preparatory expansion for interactivity

Although the application today is primarily made as a visitors simulator, it is designed in such a way that interactive features (e.g. dashboards, virtual experiments in laboratories, simple game mechanics) can be added later on.

## VI. CONCLUSION

This work addresses the challenge when capturing photos and calibrating, we encounter the problem of uneven exposures due to uneven lighting and reflections within captured photos. In order to maintain consistent input photo quality, we adjusted the camera exposure manually and followed it up with batch color correction to bring all the photos to a standardized color profile and brightness level.

Future plans for the tour would be adding settings for the graphics quality to improve performance on older devices by enabling lower quality upon start-ups.

This work showed the possible combination of two technologies to create a model usable for first-person gameplay with realistic hitboxes. Ensuring optimization also for computers with weaker performance was another problem we addressed earlier.

## ACKNOWLEDGMENT

This paper was supported by DISIC (09I05-03-V2), NEXT (ERASMUS-EDU-2023-CBHE-STRAND-2), EULiST (ERASMUS), CYB-FUT (ERASMUS+), InteRViR (VEGA 1/0605/23)

## REFERENCES

[1] Jawset, “Postshot,” Jawset. [Online]. Available: https://www.jawset.com/

[2] RadianceFields.com, “Postshot,” RadianceFields.com. [Online]. Available: https://radiancefields.com/platforms/postshot

[3] A. S. Walker and G. Petrie, “Digital Photogrammetric Workstations 1992–96,” International Archives of Photogrammetry and Remote Sensing, vol. XXXI, part B2, pp. 384–395, 1996. [Online]. Available: https://www.isprs.org/proceedings/XXXI/congress/part2/384_XXXI-part2.pdf

[4] E. Albertz and H. König, “A first concept of a digital photogrammetric workstation,” International Archives of Photogrammetry and Remote Sensing, vol. 28, part 2, pp. 262–269, 1984. [Online]. Available: https://www.asprs.org/wp-content/uploads/pers/1992journal/jan/1992_jan_51-56.pdf

[5] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, “3D Gaussian Splatting for Real-Time Radiance Field Rendering,” arXiv preprint arXiv:2308.04079, Aug. 8, 2023. [Online]. Available: https://arxiv.org/abs/2308.04079

[6] Capturing Reality, “RealityCapture,” RealityCapture. [Online]. Available: https://www.capturingreality.com/

[7] C. M. Erikson, “Hierarchical Levels of Detail to Accelerate the Rendering of Large Static and Dynamic Polygonal Environments,” Ph.D. dissertation, Dept. of Computer Science, Univ. of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 2000. [Online]. Available: https://wwwx.cs.unc.edu/~geom/papers/documents/dissertations/erikson00.pdf