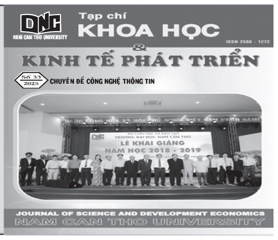
Journal of Science and Development Economics
Nam Can Tho University

Website: jsde.nctu.edu.vn



Automatic 3D reconstruction and rendering based on sparse images

Ngo Khoa Dang[1], Huynh Ngoc Thai Anh[1], Doan Hoa Minh[2], Nguyen Van Linh[2*], Ngo Truc Hung[2]

[1]Can Tho University, Can Tho City, Vietnam

[2]Nam Can Tho University, Can Tho City, Vietnam

[*]Corresponding author: Nguyễn Văn Linh (email: nguyenvanlinh@nctu.edu.vn)

Received: 20/1/2025

Revised: 3/2/2025

Accepted: 15/2/2025

Keywords: *3D rendering, 3D reconstruction, Gaussian splatting, structure from motion*

Từ khóa: *cấu trúc từ chuyển động, kết xuất ảnh 3D, phục dựng mô hình 3D, phân tán Gaussian*

ABSTRACT

3D object reconstruction and rendering have a vital role in computer vision due to its wide range of applications. There are several techniques to achieve this, including laser scanner and photogrammetry. In recent years, artificial intelligence-based methods, such as neural radiance field, have been proposed and received impressive results. This paper presents a method that applies photogrammetry for automatic 3D point reconstruction of an object, and followed by rendering using Gaussian splatting. In our work, we first capture images of an object using normal cameras. The capture path involves moving 360 degrees around the object at 3 different heights. Second, structure from the motion-based method is employed to reconstruct its 3D point cloud on the surface of the object. Next, the 3D point cloud is converted to and considered as initial 3D Gaussian points. Then, these Gaussian points are trained based on the difference between projection syntheses of Gaussian points on training images. Finally, a rendering image for a novel view is produced by projection syntheses of Gaussian splatting on that surface. Our experiments showed that this flow-chart gives superior results on 3D reconstruction and rendering. With a small object, specially a fabric flower model with the size 50cm x 40cm x 40cm, it is necessary to use approximately 90 training images, and the processing time needed to generate the 3D Gaussian point cloud for rendering is about 30 minutes.

TÓM TẮT

Phục dựng và kết xuất đối tượng 3 chiều hiện nay đang có một vai trò quan trọng trong lĩnh vực thị giác máy tính bởi những ứng dụng rộng rãi của nó. Hiện nay có nhiều kỹ thuật được áp dụng để thực hiện việc này, chẳng hạn như dùng máy quét laser hoặc kỹ thuật quang trắc *(photogrammetry). Những năm gần đây, các phương pháp dựa trên trí tuệ nhân tạo, như Neural Radiance Field, đã được đề xuất và đạt được những kết quả ấn tượng. Bài báo này trình bày một phương pháp áp dụng quang trắc để tự động phục dựng đám mây điểm 3D của một đối tượng, và sau đó sử dụng phương pháp Gaussian splatting để kết xuất hình ảnh đối tượng 3D. Trong việc này, đầu tiên, hình ảnh của đối tượng được chụp bằng các máy ảnh thông thường. Quá trình chụp bao gồm việc di chuyển 360 độ vòng quanh đối tượng và chụp ở ba độ cao khác nhau. Thứ nhì, phương pháp dựa trên sự phục dựng cấu trúc từ chuyển động (structure from motion) được sử dụng để tái tạo đám mây điểm 3D trên bề mặt đối tượng. Kế tiếp, đám mây điểm 3D này được chuyển đổi và coi như các điểm Gaussian 3D khởi đầu. Và rồi, những điểm Gaussian này sẽ được tiến hành huấn luyện để kết xuất bằng cách tối ưu hóa sự khác biệt giữa các phép tổng hợp hình chiếu (projection syntheses) của đám mây điểm Gaussian trên các hình ảnh huấn luyện. Sau cùng, hình ảnh kết xuất từ một góc nhìn bất kỳ được tạo ra bằng cách tổng hợp hình chiếu của những Gaussian trên mặt phẳng góc nhìn mới đó. Thực nghiệm của chúng tôi cho thấy rằng qui trình nguyên lý này mang lại kết quả ưu việt trong phục dựng và kết xuất hình ảnh của vật thể 3D. Với một đối tượng nhỏ, cụ thể như một mô hình bông hoa bằng chất liệu vãi có kích thước 50cm x 40cm x 40cm, số lượng hình ảnh cần thiết cho huấn luyện là vào khoảng 90 ảnh, và thời gian cần xử lý để tái tạo vật thể 3D là xấp xỉ 30 phút.*

1. INTRODUCTION

View synthesis is a task of presenting an object or a scene in any novel view from a few viewpoints captured by camera. One approach to achieve this is based on 3D geometry where an object is represented by a point cloud [1], a set of voxels [2] or of meshes [3] which are generated from images captured from different positions, as shown in Figure 1.

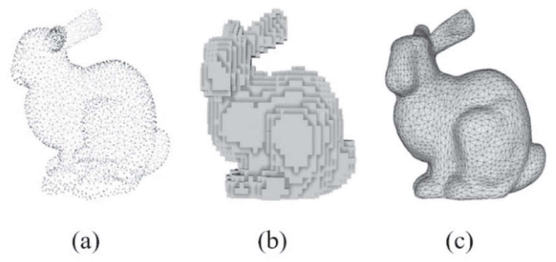


Figure 1. Presentation of a 3D object using point cloud, voxels and polygon mesh

In recent years, several learning-based methods have been proposed for novel view synthesis. These approaches have gained significant prominence. A neural radiance field (NeRF) [4] presents an object by image generated from a neural network. This technique uses spatial locations (x, y, z) and view directions in Euler angles (θ, ϕ) of pixels as inputs to train a neural network for predicting the output colors and opacity. Subsequently, several NeRF-based methods have been developed to achieve better rendering results such as MIP-NeRF [5], MIP-Nerf 360 [6]. In 2023, a new technique was introduced in which an object was rendered by 3D Gaussian splatting [7]. This method shows that it improves both rendering quality and training time.

This paper presents a method to automatically reconstruct a 3D point cloud model, and employs Gaussian splatting to render 3D object using images or videos captured by normal cameras. It helps to reduce manual work in 3D object reconstruction and rendering.

2. RESEARCH METHODS

Our method consists of three main steps: discovering methods, collecting data, and proposing and applying an effective method for 3D object reconstruction and rendering. We started from methods which use mesh-based rendering to the most up-to-date approaches such as neuron-based ones. Then, we focused on 3D Gaussian-splatting. In our work, a variety of cameras was used for data collection. After that, we installed open sources and support free-ware which are cited and downloaded from Github. Finally, our developed code is deployed on a computer with GPU Nvidia RTX 3090Ti, 64GB Ram for training and testing.

3. RESULTS AND DISCUSSION

3.1 Gaussian splatting

3D Gaussian splatting (GS) [7] is a rasterization technique which converts 3D Gaussian points in 3D space for photorealistic rendering. The difference from a mesh-based representation is that instead of using polygon mesh which is created from at least 3 points (triangular mesh), the GS-based technique uses Gaussian. To present an object, the mesh-based method uses a set of polygons, while 3D-GS does a set of Gaussian points as depicted in Figure 2 and Figure 3.

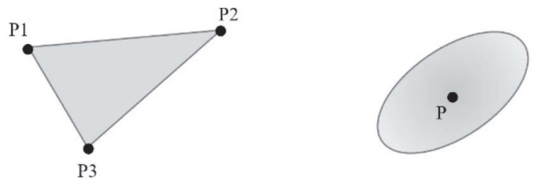


Figure 2. Presentation of an area of an object. Left image and right images present an area of an object using triangular and Gaussian, respectively.

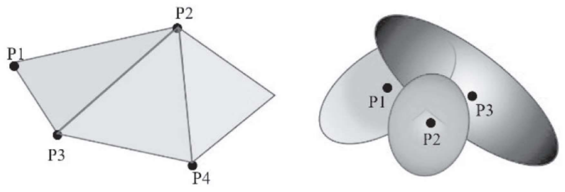


Figure 3. Presentation of an area of object by combining of triangle mesh (left image) and 3D Gaussian (right image)

A Gaussian point is defined as in Eq. (1)

$$G(x|\mu,\Sigma) = e^{-\frac{1}{2}(x-\mu)^{\mathrm{T}}\Sigma^{-1}(x-\mu)} \quad (1)$$

where $\mu \in \mathrm{R}^3$ represents the spatial mean and $\boldsymbol{\Sigma} \in \mathrm{R}^{3x3}$ denotes the covariance matrix. To ensure validity throughout the optimization process, the covariance matrix $\boldsymbol{\Sigma}$ is decomposed into a scaling matrix $\boldsymbol{S}$ and a rotation matrix $\boldsymbol{R}$ as Eq. (2).

$$\Sigma = RSS^{\mathrm{T}}R^{\mathrm{T}} \quad (2)$$

During the rendering process, the 3D Gaussians are projected onto a 2D plane of camera. Given a viewing transformation W, the covariance matrix Σ' in camera coordinates is given as Eq. (3).

$$\Sigma' = JW\Sigma W^{T}J^{T} \quad (3)$$

where $\boldsymbol{J}$ represents the Jacobian of the affine approximation of the projective transformation. Each

Gaussian is associated with an opacity value σ and a view-dependent color $\boldsymbol{c}$, determined by a set of spherical harmonics coefficients. The pixel color C is computed by performing α-blending on the sorted 2D Gaussians, starting from the front and progressing toward the back, as shown in Eq. (4):

$$C = \sum_{i\in N} T_i G_i\,(\mu \mid \mu', \Sigma')\sigma_i c_i, \tag{4}$$

where $T_i = \prod_{j=1}^{i-1}\left(1 - \sum_{i\in N} T_i G_i\,(\mu \mid \mu', \Sigma'')\sigma_i\right)$

3.2 Optimization

The optimization is based on a comparison between the rendering images with the training images. This process is iterated on all images in captured dataset. To optimize, the derivatives of scale and rotation of each Gaussian point need to calculate. Assume that Σ/Σ' are the world/view space covariance matrices of the Gaussian, q is the rotation, and s the scaling, W is the viewing transformation and J the Jacobian of the affine approximation of the projective transformation. The derivatives w.r.t scaling and rotation are calculated as Eq. (5) and Eq. (6) using the chain rule.

$$\frac{\mathrm{d}\Sigma'}{\mathrm{d}s} = \frac{\mathrm{d}\Sigma'}{\mathrm{d}\Sigma}\frac{\mathrm{d}\Sigma}{\mathrm{d}s} \tag{5}$$

$$\frac{\mathrm{d}\Sigma'}{\mathrm{d}q} = \frac{\mathrm{d}\Sigma'}{\mathrm{d}\Sigma}\frac{\mathrm{d}\Sigma}{\mathrm{d}q} \tag{6}$$

where $\frac{\partial\Sigma'}{\partial\Sigma_{ij}} = \begin{pmatrix} U_{1,i}U_{1,j} & U_{1,i}U_{2,j} \\ U_{1,j}U_{2,i} & U_{2,i}U_{2,j} \end{pmatrix}$, $U = JW$,

$\frac{d\Sigma}{ds} = \frac{d\Sigma}{dM}\frac{dM}{ds}$, $\frac{d\Sigma}{dq} = \frac{d\Sigma}{dM}\frac{dM}{dq}$, $M = RS$, $\frac{d\Sigma}{dM} = 2M^T$, and

For scaling: $\frac{\partial M_{i,j}}{\partial s_k} = \begin{cases} R_{i,k} & \text{if } j = k \\ 0 & \text{otherwise} \end{cases}$

For rotation, we recall the conversion from a unit quaternion q with real part q_r and imaginary parts q_i, q_j, q_k to a rotation matrix R:

$$R(q) = 2\begin{pmatrix} \frac{1}{2} - \left(q_j^2 + q_k^2\right) & \left(q_iq_j - q_rq_k\right) & \left(q_iq_k - q_rq_j\right) \\ \left(q_iq_j - q_rq_k\right) & \frac{1}{2} - \left(q_i^2 + q_k^2\right)0 & \left(q_jq_k - q_rq_i\right) \\ \left(q_iq_k - q_rq_j\right) & \left(q_jq_k + q_rq_i\right) & \frac{1}{2} - \left(q_i^2 + q_j^2\right) \end{pmatrix} \tag{7}$$

Then, the gradients for the components of q are calculated as Eq. (8), Eq. (9), Eq. (10), Eq. (11). Deriving gradients for quaternion normalization is straightforward.

$$\frac{\partial M}{\partial q_r} = 2\begin{pmatrix} 0 & -s_yq_k & s_zq_j \\ s_xq_k & 0 & -s_zq_i \\ -s_xq_j & s_yq_i & 0 \end{pmatrix} \tag{8}$$

$$\frac{\partial M}{\partial q_i} = 2\begin{pmatrix} 0 & -s_yq_j & s_zq_k \\ s_xq_j & -2s_yq_i & -s_zq_r \\ -s_xq_k & s_yq_r & -2s_zq_i \end{pmatrix} \tag{9}$$

$$\frac{\partial M}{\partial q_j} = 2\begin{pmatrix} -2s_xq_j & s_yq_i & s_zq_r \\ s_xq_i & 0 & s_zq_k \\ -s_xq_r & s_yq_k & -2s_zq_j \end{pmatrix} \tag{10}$$

$$\frac{\partial M}{\partial q_k} = 2\begin{pmatrix} -2s_xq_k & -s_yq_r & s_zq_i \\ s_xq_r & -2s_yq_k & -s_zq_j \\ s_xq_i & s_yq_j & 0 \end{pmatrix} \tag{11}$$

The optimization for getting good renders includes 3 components: good initialization, differentiable optimization, and adaptive densification. For the initial point cloud, we can use any methods to reconstruct the 3D point cloud of the object. A well-known method which is usually used for automatic reconstruction is structure from motion [8],[9]. Then, the process of differentiable optimization and adaptive densification are described as Algorithm 1, in which the densification is a process of splitting a point becoming 2 points if the size of the Gaussian point is large.

```
Algorithm 1: Optimization and Densification
$w$, $h$: width and height of the training images
$M$ ← initial points
$S, C, A$ ← Initialize attributes()
$i$ ← 0
  while not converged do
  $V, \hat{I}$ ← SampleTrainingView()
  $I$ ← Rasterize($M, S, C, A, V$ )
  $L$ ← $Loss(I, \hat{I})$
  $M, S, C, A$ ← Adam($\nabla L$)
```

```
    if IsRefinementIteration($i$) then for all
      Gaussians ($\mu$, $\Sigma$, $c$, $\alpha$) in ($M$, $S$, $C$, $A$) do
        if $\alpha < \epsilon$ or IsTooLarge($\mu$, $\Sigma$) then
          RemoveGaussian()
        end if
        if $\nabla pL > \tau_p$ then
          if $||S|| > \tau_S$ then
            SplitGaussian($\mu$, $\Sigma$, $c$, $\alpha$)
          Else
            CloneGaussian($\mu$, $\Sigma$, $c$, $\alpha$)
          End if
        End if
      End for
    End if
    $i \leftarrow i + 1$
  End while
```

3.3 Block diagram of system

Figure 4 presents the flow chart for novel view synthesis of an object in this work. First, the images are acquired from any cameras. Second, the poses and positions of images are estimated. Next, the point cloud of object is obtained by using any methods for 3D reconstruction. These points are considered as initial points for training. The training process uses the point cloud, render in a specific pose and location of an input training image. Then, the difference is used to update Gaussian point parameters. The training finishes when the number of steps is met. For the render phase, an image in novel view is obtained by synthesis of projected trained Gaussian points on a specific pose and location.

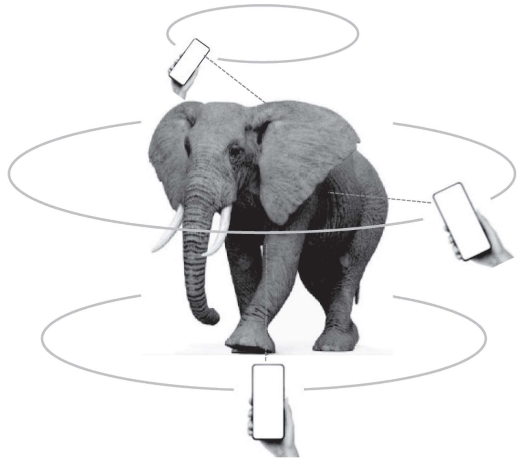
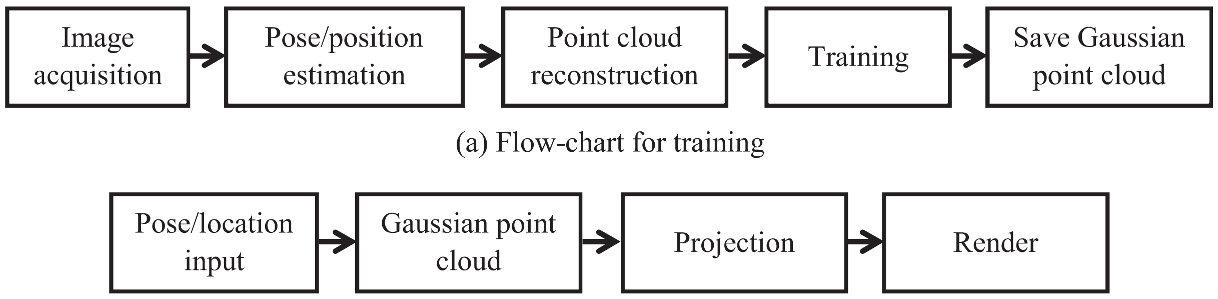


Figure 4. View point path planning for object capture



(a) Flow-chart for training

(b) Flow-chart for rendering

Figure 5. Flow-chart for training and rendering of an object from sparse images

3.4 Experiment

3.4.1 Experiment setting up

In our experiments, first, a short video of an object was captured for image retrieval using hand phone camera. To get better result for rendering an object, we captured 3 rounds of 360 degrees surrounding the object, one on the top, one the middle, and the last one on the bottom, as shown in Fig. 5. Second, the images were obtained by extracting frames from the video with 1 frame per second. A requirement in this step is that an extracted image must have overlap regions with its neighbors. The overlap regions have to be sufficiently large so that they can use for pose and location estimation. In our experiments, overlap areas are around 70% with sampling 1 frame per second. Next, the structure from motion method is applied to estimate camera poses and positions of extracted frames automatically. Then, the point cloud for the object was generated based on matching points in multi-view. Finally, this point cloud was converted to and considered as an initial Gaussian point cloud for training. After training completed, it can be used to render the object at any novel views. An interactive viewer was utilized to capture a position and rotation in the real-world coordinate system, and used as inputs to render an output image of the object.

3.4.2 Experimental results

In this paper, we did experiments on 2 objects: an elephant toy and a flower model. First, we used an Iphone to record a video of the elephant model in 65 seconds, and the flower in 93 seconds. Second, frames of the video were extracted with 1 frame per second. Figure 6 and Fig. 7 presents some images of the elephant and flower models. The structure from motion method was used for automatic camera pose estimation. Figure 8(a) and 8(b) show the camera poses of extracted images for elephant and flower models, respectively.

After training completed, Gaussian points are used for rendering. Figure 9 and Fig. 13 present visual comparison between a rendering view and its ground truth image. Figure 10 and Fig. 14 show rendering images in arbitrary novel views. To see the affection of Gaussian scale, we presented rendering images in different Gaussian scales as shown in Fig. 11 and Fig. 15. Finally, the rendering model in novel views is shown in Fig. 12 and Fig. 16. Comparing between the input images and the rendering images visually, it can be seen that Gaussian splatting is a good technique to present the 3D objects. These rendering scenes can be embedded to a game engine for further applications. Besides visual evaluation, we also recorded consuming time for the whole process.

With about 60 and 90 images of elephant and flower models, respectively, the processing time is around 20 minutes and 30 minutes.

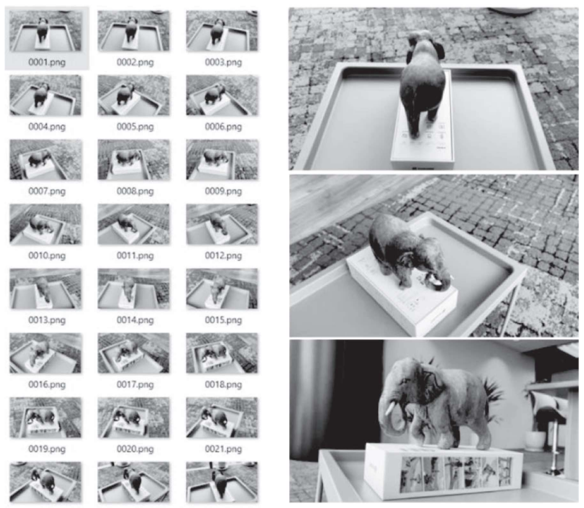


Figure 6. Captured images of the elephant model

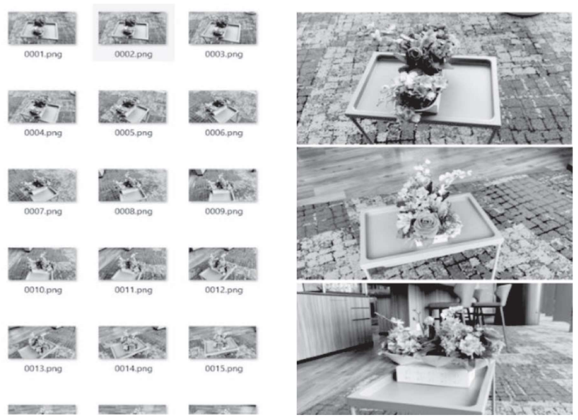


Figure 7. Captured images of the flower model

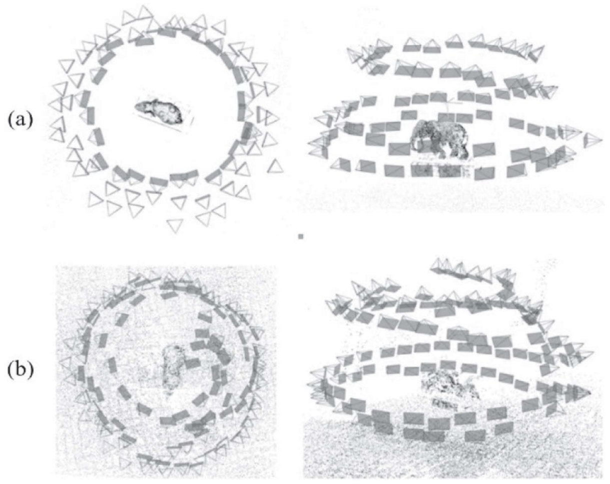


Figure 8. Estimation of camera poses and positions. ***(a) for elephant model. (b) for flower model***

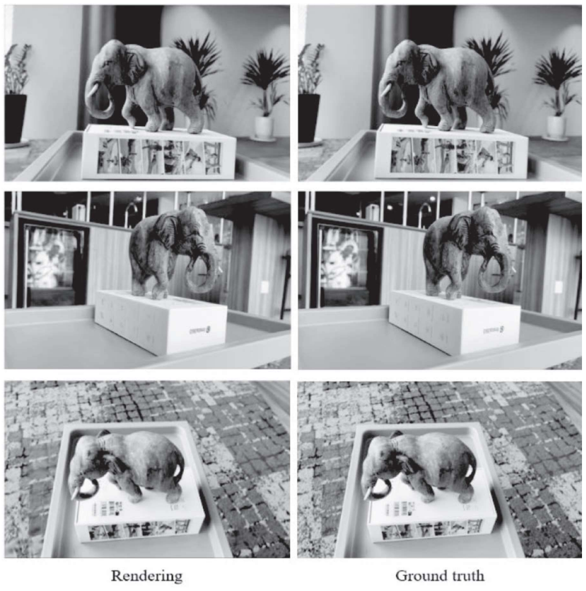


Figure 9. Visual comparison between rendering and input images of elephant model

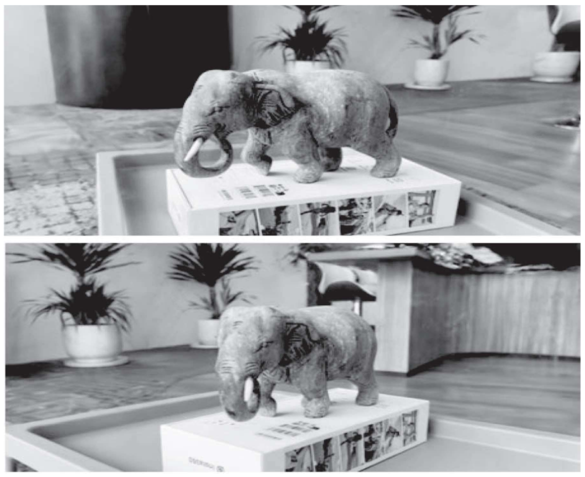
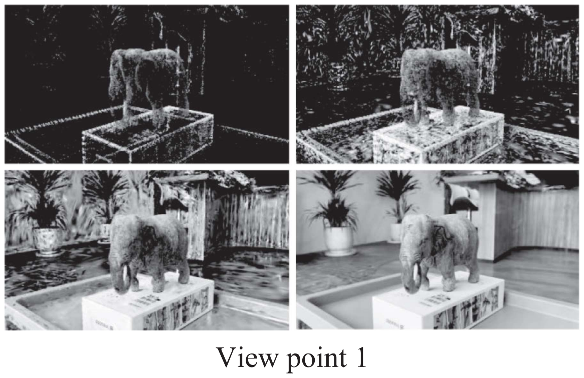


Figure 10. Rendering images in novel viewpoints of elephant model



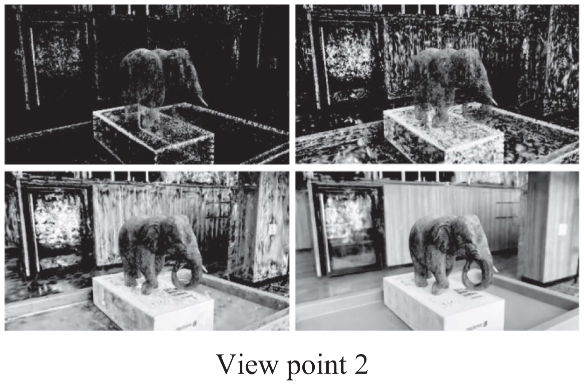


Figure 11. Rendering images of elephant model in different Gaussian scales. From left to right, top to bottom, the scales are 0.01, 0.2, 0.5, and 1.0, respectively

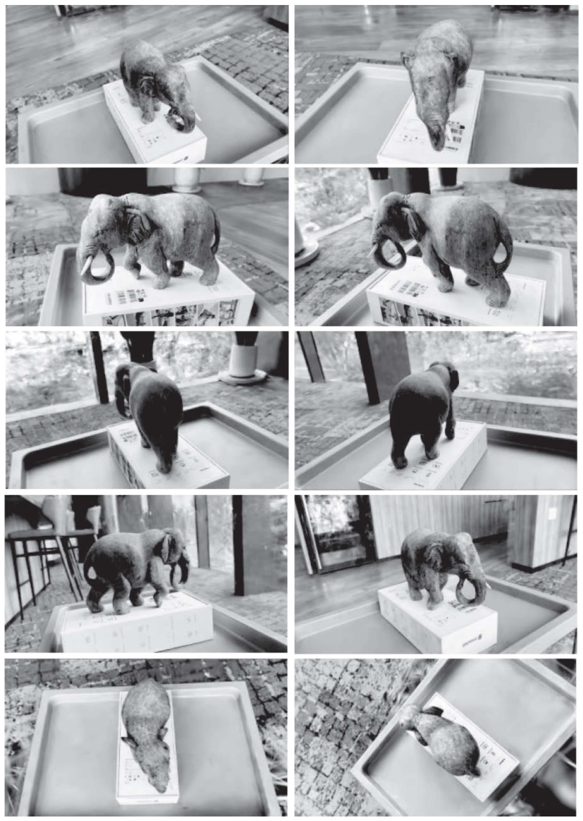


Figure 12. Rendering images from novel viewpoints of elephant model

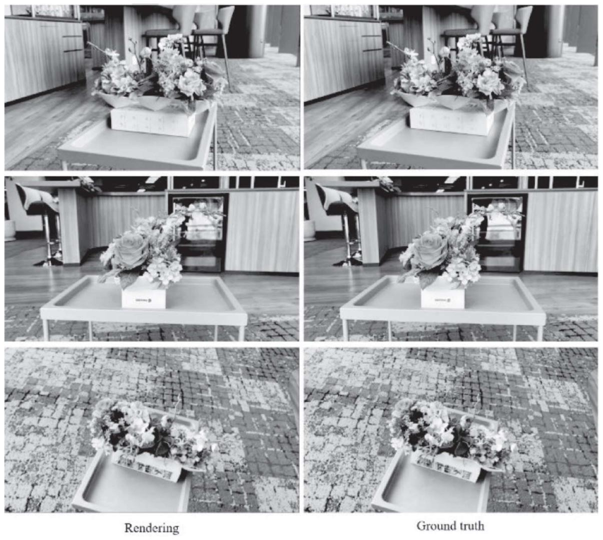


Figure 13. Visual comparison between rendering and input images of flower model

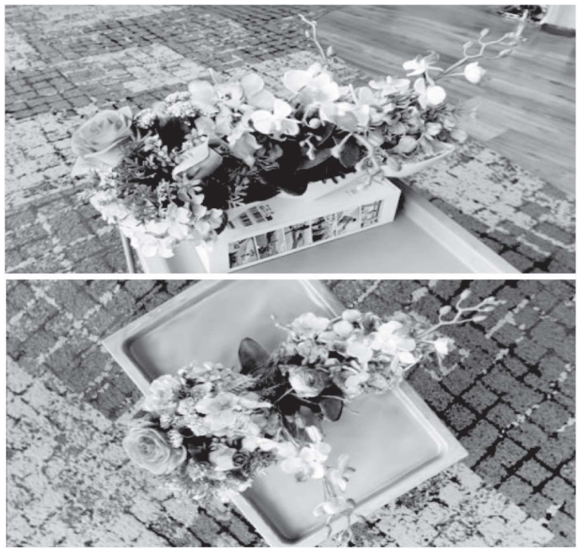
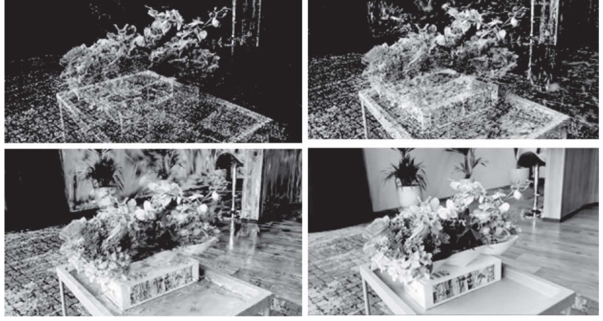
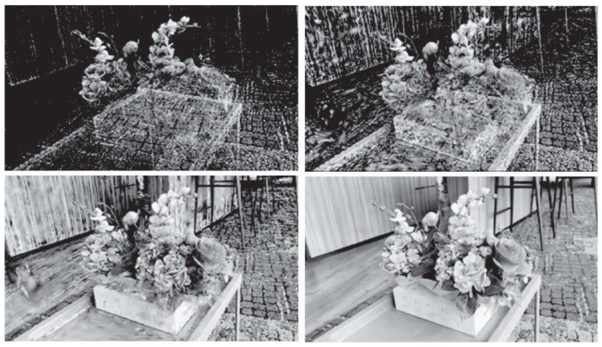


Figure 14. Rendering images in novel viewpoints of flower model



Viewpoint 1



Viewpoint 2

Figure 15. Rendering images of flower model in different Gaussian scales. From left to right, top to bottom, the scales are 0.01, 0.2, 0.5, and 1.0, respectively

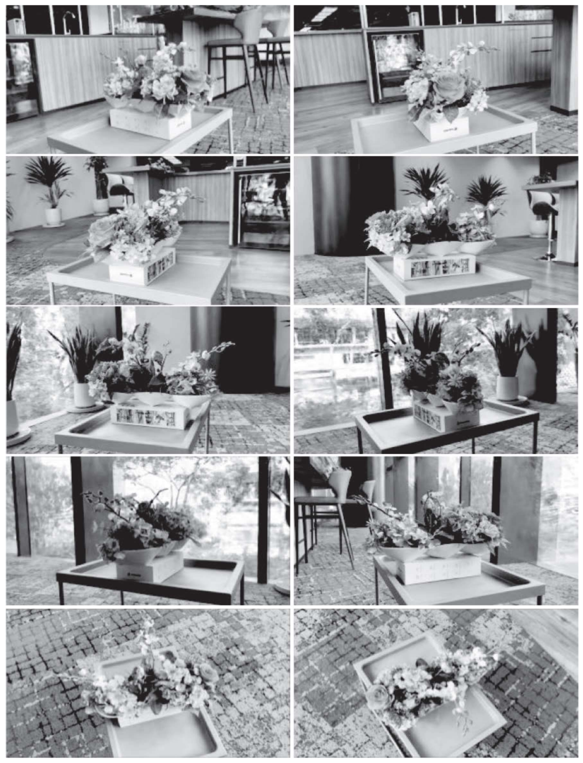


Figure 16. Rendering images from novel viewpoints of flower model

4. CONCLUSION

This paper presents a technique for automatic 3D reconstruction and rendering of an object using structure from motion and Gaussian splatting. In our work, we used normal and cheap cameras such as phone cameras. This helps people able to reconstruct their interesting models using their devices. From experiments, it shows that this method gives a good quality for visualization.

Therefore, it is suitable to use in some applications that require high quality. With consuming time for the whole process to be around some minutes, this technique makes it become more practical for model reconstruction. In addition, the rendering technique is based on projection of Gaussian, it does not require devices with high configuration. The viewer can even run on webGL for rendering. With the above points, this method can be deployed in many applications in the future.

REFERENCES

[1] Choy, C. B., Xu, D., Gwak, J., Chen, K., & Savarese, S. (2016). 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII 14* (pp. 628-644). Springer International Publishing.

[2] Fishman, E. K., Ney, D. R., Heath, D. G., Corl, F. M., Horton, K. M., & Johnson, P. T. (2006). Volume rendering versus maximum intensity projection in CT angiography: what works best, when, and why. *Radiographics*, *26*(3), 905-922.

[3] Tobler, R. F. (2006). A mesh data structure for rendering and subdivision.

[4] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., & Ng, R. (2021). Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, *65*(1), 99-106.

[5] Jonathan T. B., Ben M., Matthew T., Peter H., Ricardo M. B., Pratul P. S. (2021), "Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields", Proceedings of Computer Vision and Pattern Recognition, pp. 5855–5864.

[6] Jonathan T. B., Ben M., Dor V., Pratul P. S., Peter H. (2022), "Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields", *Proceedings of Computer Vision and Pattern Recognition*, pp. 5470–5479.

[7] Kerbl, B., Kopanas, G., Leimkühler, T., & Drettakis, G. (2023). 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, *42*(4), 139-1.

[8] Ullman, S. (1979). The interpretation of structure from motion. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, *203*(1153), 405-426.

[9] Linda G. S., George C. S. (2001). *Computer Vision*. Prentice Hall.