SurfaceGS: Dynamic Surface Gaussian Splatting for Urban Driving Scenes

Supplementary Material

1. Overview

We begin by outlining the structure of this supplementary material. Sec. 2 provides an extended discussion of different surface representations, offering deeper insights into the trade-offs between Bézier surfaces, B-spline surfaces, Hermite surfaces and implicit formulations. Sec. 3 elaborates on the formulation of the Bézier curve used for global motion modeling. Sec. 4 presents additional explanations of our loss function. Sec. 5 details our implementation details. Sec. 6 offers more experiments, including several hyperparameter studies that validate the robustness of our design choices. Sec. 7 provides more visualizations, showcasing qualitative comparisons that highlight the advantages of our method.

2. Different Surface Representations

In urban driving scenes, surface representations must capture rigid vehicle bodies as well as non-rigid pedestrians under a unified formulation. In this section, we provide an overview of several representative surface formulation, including Bézier, B-spline, NURBS, Hermite, and analytic surfaces. All parametric surfaces are denoted as

$$S_\star(u,v) : [0,1]^2 \to \mathbb{R}^3, \tag{1}$$

where the subscript $\star \in \{bez, \mathrm{bs}, nrb, her, ana\}$ indicates the specific representation.

2.1. Bézier Surface

A tensor-product Bézier surface of degrees (n_u, n_v) is defined as

$$S_{bez}(u,v) = \sum_{i=0}^{n_u}\sum_{j=0}^{n_v} B_{i,n_u}(u)\, B_{j,n_v}(v)\, \boldsymbol{P}_{ij}, \tag{2}$$

where $\boldsymbol{P}_{ij} \in \mathbb{R}^3$ are control points and $B_{i,n}(u)$ are Bernstein basis polynomials

$$B_i^n(u) = \binom{n}{i} u^i (1-u)^{n-i}, \quad i = 0, \dots, n. \tag{3}$$

Bézier surfaces are C^∞-smooth inside the parameter domain and admit stable differentiation. However, each control point affects the entire patch, making local refinement and highly localized deformation more difficult.

2.2. B-spline Surface

A tensor-product B-spline surface uses locally supported basis functions:

$$S_{bs}(u,v) = \sum_{i=0}^{n_u}\sum_{j=0}^{n_v} N_{i,p}(u)\, M_{j,q}(v)\, \boldsymbol{P}_{ij}, \tag{4}$$

where $N_{i,p}(u)$ and $M_{j,q}(v)$ are univariate B-spline basis functions of degrees p and q defined on knot vectors $U = \{u_0, \dots, u_{n_u+p+1}\}$ and $V = \{v_0, \dots, v_{n_v+q+1}\}$. The basis functions satisfy the recursion

$$N_{i,0}(u) = \begin{cases} 1, & u_i \le u < u_{i+1}, \\ 0, & \text{otherwise}, \end{cases} \tag{5}$$

$$\begin{aligned} N_{i,p}(u) = {} & \frac{u - u_i}{u_{i+p} - u_i}\, N_{i,p-1}(u) \\ & + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}}\, N_{i+1,p-1}(u). \end{aligned} \tag{6}$$

with an analogous definition for $M_{j,q}(v)$. Due to compact support, perturbing a single $\boldsymbol{P}_{ij}$ only affects a local neighborhood. Knot multiplicity controls inter-patch continuity (C^0, C^1, *etc.*), which makes B-splines well suited for complex geometries requiring local refinement, at the expense of managing both control points and knot vectors.

2.3. NURBS Surface

Non-Uniform Rational B-Splines (NURBS) extend B-spline surfaces by attaching positive weights $w_{ij} > 0$ to each control point:

$$S_{nrb}(u,v) = \frac{\sum_{i=0}^{n_u}\sum_{j=0}^{n_v} N_{i,p}(u)\, M_{j,q}(v)\, w_{ij}\, \boldsymbol{P}_{ij}}{\sum_{i=0}^{n_u}\sum_{j=0}^{n_v} N_{i,p}(u)\, M_{j,q}(v)\, w_{ij}}. \tag{7}$$

When all $w_{ij} = 1$, NURBS reduce to B-spline surfaces. The rational form enables exact modeling of circles, cylinders, spheres, and other quadrics, which is attractive for rigid man-made components. However, the denominator introduces extra nonlinearity and potential numerical instability in gradient-based optimization, and the combined parameter space of $\boldsymbol{P}_{ij}$, knot vectors, and w_{ij} can be high-dimensional in large-scale dynamic scenes.

2.4. Hermite Surface

Hermite surfaces incorporate both positions and first-order derivatives. A bicubic Hermite surface is written as

$$S_{her}(u,v) = \sum_{i=0}^{1}\sum_{j=0}^{1}\Big(h_i(u)h_j(v)\,\boldsymbol{P}_{ij} + h_i(u)\,\dot{h}_j(v)\,\boldsymbol{T}^{v}_{ij} + \dot{h}_i(u)h_j(v)\,\boldsymbol{T}^{u}_{ij} + \dot{h}_i(u)\,\dot{h}_j(v)\,\boldsymbol{T}^{uv}_{ij}\Big). \tag{8}$$

where $\boldsymbol{P}_{ij}$ are corner positions, $\boldsymbol{T}^{u}_{ij}$ and $\boldsymbol{T}^{v}_{ij}$ are tangents along the u- and v-directions, and $\boldsymbol{T}^{uv}_{ij}$ encodes mixed partial derivatives. The cubic Hermite basis on $t \in [0,1]$ is given by

$$\begin{aligned} h_0(t) &= 2t^3 - 3t^2 + 1, \quad h_1(t) = -2t^3 + 3t^2, \\ h_2(t) &= t^3 - 2t^2 + t, \qquad h_3(t) = t^3 - t^2, \end{aligned} \tag{9}$$

with derivatives $\dot{h}_k(t) = \frac{dh_k(t)}{dt}$. This representation offers direct control over first-order derivatives and facilitates smooth patch joining, but requires additional tangent and mixed-derivative data, which increases the parameter count.

2.5. Analytic Surface

Analytic surfaces are defined either implicitly or parametrically. An implicit analytic surface is written as

$$F_{ana}(x,y,z) = 0, \tag{10}$$

while a parametric analytic surface uses

$$S_{ana}(u,v) = \big(x_{ana}(u,v),\, y_{ana}(u,v),\, z_{ana}(u,v)\big)^{\top}. \tag{11}$$

A canonical example is the sphere of radius r,

$$x^2 + y^2 + z^2 = r^2, \tag{12}$$

with the parametric form

$$S_{ana}(u,v) = \begin{bmatrix} r\sin v\cos u \\ r\sin v\sin u \\ r\cos v \end{bmatrix}, \quad u \in [0, 2\pi),\ v \in [0,\pi]. \tag{13}$$

For such surfaces, normals and curvatures often admit closed-form expressions, yielding high efficiency and numerical stability. They are suited for simple rigid components or as geometric priors, but lack flexibility to capture complex, spatially varying shapes and fine-scale details.

3. Bézier Curve

A Bézier curve of degree n is a polynomial parametrization

$$C_{bez}(t) = \sum_{i=0}^{n} W_{i,n}(t)\,\boldsymbol{P}_i, \quad t \in [0,1], \tag{14}$$

Number	PSNR ↑	SSIM ↑
7×5	31.41	0.922
8×6	**31.68**	**0.931**
9×7	<u>31.66</u>	**0.931**

(a) Control Point Number.

Number	PSNR ↑	SSIM ↑
4	31.59	0.922
5	**31.68**	**0.931**
6	<u>31.66</u>	<u>0.930</u>

(b) Surface Patch Number.

Bound s_{max}	Bound s_{min}	PSNR ↑	SSIM ↑	FPS ↑
$4e-2$	$4e-7$	**31.74**	**0.935**	51
$5e-2$	$5e-7$	<u>31.68</u>	<u>0.931</u>	<u>63</u>
$6e-2$	$6e-7$	31.45	0.926	**69**

(c) Gaussian Scaling Bound.

Sensitivity α_u	Sensitivity α_v	Sensitivity β	PSNR ↑	SSIM ↑	FPS ↑
0.4	0.4	0.1	31.63	<u>0.928</u>	**64**
0.5	0.5	0.2	**31.68**	**0.931**	<u>63</u>
0.6	0.6	0.3	<u>31.65</u>	0.926	<u>63</u>

(d) Curvature Sensitivity.

Table 1. Ablation of Hyperparameters.

where $\boldsymbol{P}_i \in \mathbb{R}^3$ are control points and $W_{i,n}(t)$ are the Bernstein basis polynomials

$$W_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}, \quad i = 0,\ldots,n. \tag{15}$$

Similar to Bézier surfaces, each control point globally influences the entire curve segment, which yields intuitive shape manipulation via the control polygon but limits strict local refinement. In our context, Bézier curves naturally serve as low-dimensional trajectories (*e.g.*, object-level centerlines), while B-spline surfaces capture residual variations in space.

4. Loss Function

To jointly optimize all learnable parameters, we adopt the composite loss function defined in Eq. 16:

$$\begin{aligned} \mathcal{L} =& (1-\lambda_r)\mathcal{L}_1 + \lambda_r\mathcal{L}_{ssim} + \lambda_l\mathcal{L}_l + \\ & \lambda_s\mathcal{L}_s + \lambda_{orth}\mathcal{L}_{orth} + \lambda_{dr}\mathcal{L}_{dr} + \lambda_{dv}\mathcal{L}_{dv} \end{aligned} \tag{16}$$

which consists of several components as follows.

Image RGB Loss. $\mathcal{L}_1$ and $\mathcal{L}_{ssim}$ are L1 and SSIM losses for supervising RGB rendering, respectively.

Lidar Depth Loss. This loss is computed in the inverse-depth domain to align with projected sparse LiDAR inputs:

$$\mathcal{L}_l = \sum \left\| \boldsymbol{D}_l - \hat{\boldsymbol{D}} \right\|_1, \tag{17}$$

where $\boldsymbol{D}_l$ is the inverse sparse LiDAR map projected to the image plane, and $\hat{\boldsymbol{D}}$ is the inverse of predicted depth map.

Sky Loss. To suppress unnecessary foreground opacity in the sky and promote confident rendering, we define:

$$\mathcal{L}_s = -\sum \boldsymbol{M}_{sky} \cdot \log(1 - \boldsymbol{O}) \tag{18}$$

where $\boldsymbol{O}$ is the rendered opacity map, and $\boldsymbol{M}_{sky}$ is a binary sky mask predicted by Grounded-SAM [3, 6]. This loss penalizes nonzero opacity in sky regions, ensuring the sky is rendered through background texture without interference from foreground geometry.

Orthogonality Loss. To further ensure numerical stability and decorrelation of the basis matrices, we introduce a soft orthogonality constraint during training, as shown in the main paper.

Dynamic Rendering Loss. Following [4], to prevent mutual interference between dynamic and static Gaussian primitives during alpha blending, we enforce that dynamic regions are rendered exclusively by dynamic Gaussians. Accurate dynamic masks $\boldsymbol{M}_d$ are obtained by projecting dynamic 3D bounding boxes onto the image plane and refining them with Grounded-SAM. These masks supervise the RGB reconstruction of dynamic Gaussians through masked L1 and SSIM losses:

$$\mathcal{L}^d_{rgb} = (1 - \lambda_r)\mathcal{L}^d_1 + \lambda_r \mathcal{L}^d_{ssim}. \tag{19}$$

To further align the rendered opacity of dynamic Gaussians with the true dynamic regions, we introduce an alpha-matching term:

$$\mathcal{L}^d_o = \left\| \boldsymbol{M}_d - \boldsymbol{O}_d \right\|_1. \tag{20}$$

The final dynamic rendering loss is defined as:

$$\mathcal{L}_{dr} = \mathcal{L}^d_{rgb} + \mathcal{L}^d_o. \tag{21}$$

This loss enhances the separation between dynamic and static components and improves rendering quality under novel viewpoints.

Dynamic Velocity Loss. Following [4], to ensure that the motion of dynamic Gaussian primitives remains consistent with that of the underlying dynamic objects, we impose an additional velocity constraint. Since both the object center trajectory $\boldsymbol{\xi}(h, t)$ and the offset $\boldsymbol{\eta}(t)$ are modeled using Bézier curves and B-spline surfaces, their velocities can be computed analytically from the derivatives. The resulting instantaneous velocity $\boldsymbol{v}(h, \tau)$ is used to render the velocity map $\boldsymbol{V}$. To prevent dynamic primitives from drifting into static regions, we restrict their motion using the dynamic mask $\boldsymbol{M}_d$ and define the velocity loss as:

$$\mathcal{L}_{dv} = \left\| \boldsymbol{V} \cdot (1 - \boldsymbol{M}_d) \right\|_2. \tag{22}$$

By reasonably constraining the velocity field, the movement of dynamic Gaussians is confined within the dynamic objects, thereby improving the reliability and physical plausibility of the dynamic representation.

5. Implementation Details

We use B-spline surfaces with $m = 7, n = 5, p = 5, n = 3$, and adopt a standard cubic Bézier curve. Uniform B-spline basis functions are implemented using a matrix formulation [2, 5] for efficient computation. For Gaussian scaling, we set $s_{max} = 0.05$, $s_{min} = 5e - 7$, $\alpha_u = 0.5$, $\alpha_v = 0.5$, $\beta = 0.2$, $\lambda = 0.05$. The Tucker rank is set to $r_u = 5$, $r_v = 3$, $r_t = 3$. The number of surfaces assigned to each object is set to $K_c = 5$. The loss weights λ_r, λ_l, λ_s, λ_{orth}, λ_{dr}, λ_{dv} are set to 0.2, 1.0, 0.05, 0.01, 0.1, 1.0.

For initialization, we construct a hybrid control-point set by combining 6×10^5 LiDAR points with 2×10^5 auxiliary samples drawn from a spherical distribution. Following [1], the radius of each sampled point is generated through a reciprocal interpolation, while the angular components follow non-uniform distributions that emphasize the upper hemisphere. The resulting spherical samples are concatenated with the LiDAR points to form the final initialization set. For training, we use Adam optimizer, and the model is trained for a total of 30,000 iterations with all parameters optimized jointly. All experiments are conducted on a single NVIDIA A100 GPU.

6. More Experiments

We examine the influence of different hyperparameters on SurfaceGS in Sec. 6.1.

6.1. Hyperparameter

We conduct a ablation study on several hyperparameters of SurfaceGS, including the number of control points, the number of surface patches, the Gaussian scale bounds, and the curvature sensitivity, with results summarized in Tab. 1.

Control Point Number. Tab. 1(a) evaluates the effect of different control point configurations. An insufficient number of control points restricts the expressiveness of the dynamic surface and fails to capture local deformations, whereas an excessive number introduces redundant degrees of freedom and increases susceptibility to noise. A moderate configuration (8×6) provides the best balance between smoothness and local fitting ability.

Surface Patch Number. As shown in Tab. 1(b), increasing the number of patches enhances the model's ability to capture local geometric variations but simultaneously increases the number of control points per region, introducing redundant degrees of freedom and making the optimization more sensitive to noise. Setting the number of patches to five provides the best balance in dynamic vehicle scenes.

Gaussian Scale Bounds. Tab. 1(c) investigates the impact of the upper and lower bounds of Gaussian scales. A smaller lower bound prevents Gaussian degeneration, whereas an overly large upper bound produces excessive smoothing and reduces fine-detail fidelity. The configuration $(s_{max}, s_{min}) = (5e - 2, 5e - 7)$ achieves the best trade-off among PSNR, SSIM, and rendering efficiency.

Curvature Sensitivity. Tab. 1(d) analyzes the curvature-guided anisotropic scaling coefficients. Low sensitivity fails

Figure 1. More visualization results for image reconstruction.

Figure 2. More visualization results for novel view synthesis.

to eflect local surface geometry, while overly high sensitivity amplifies noise and destabilizes the geometric structure. A moderate sensitivity setting $(0.5, 0.5, 0.2)$ achieves the optimal rendering performance.

7. More Visualizations

We provide more visualization results for image reconstruction and novel view synthesis on the Waymo dataset in Fig. 1, Fig. 2, Fig. 3 and Fig. 4, further demonstrating the ability to improve rendering fidelity.

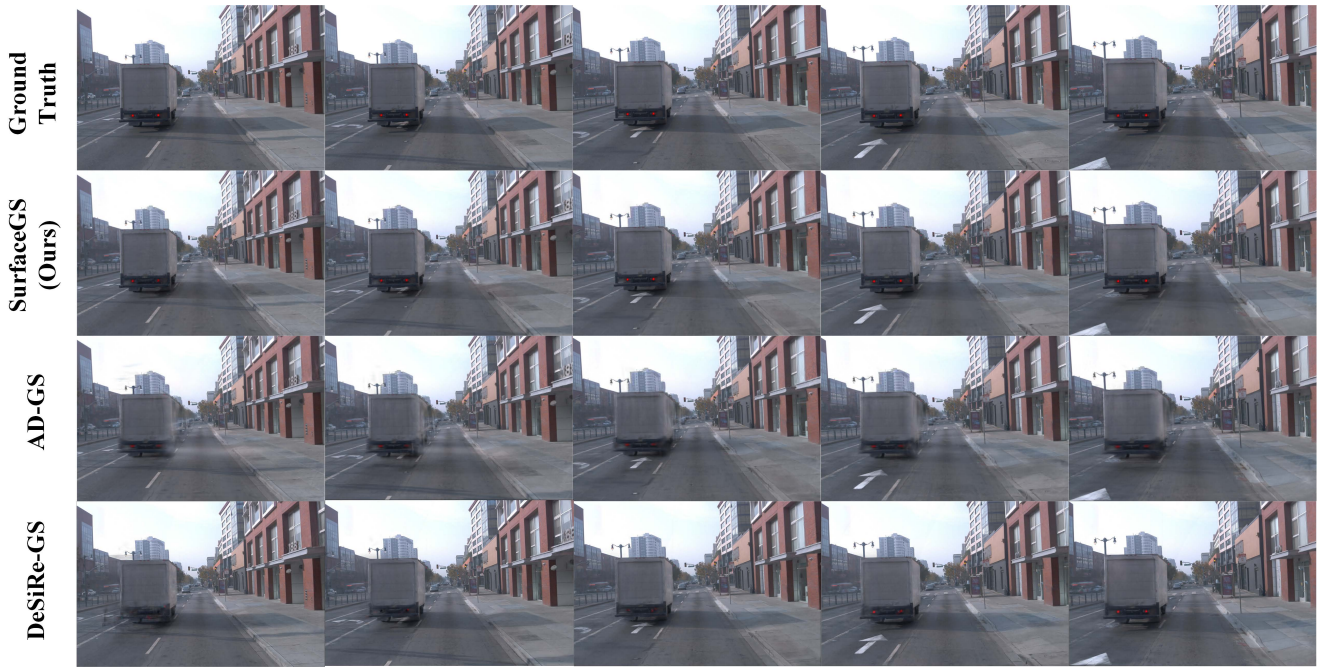

Figure 3. More visualization results for novel view synthesis.

Figure 4. More visualization results for novel view synthesis.

References

[1] Yurui Chen, Chun Gu, Junzhe Jiang, Xiatian Zhu, and Li Zhang. Periodic vibration gaussian: Dynamic urban scene reconstruction and real-time rendering. *arXiv preprint*, 2023. 3

[2] Xu Jiawei, Deng Kai, Fan Zexin, Wang Shenlong, Xie Jin, and Yang Jian. Ad-gs: Object-aware b-spline gaussian splatting for self-supervised autonomous driving. In *ICCV*, 2025. 3

[3] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer White-

head, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *ICCV*, 2023. 3

[4] Zipei Ma, Junzhe Jiang, Yurui Chen, and Li Zhang. Béziergs: Dynamic urban scene reconstruction with bézier curve gaussian splatting. In *ICCV*, 2025. 3

[5] Kaihuai Qin. General matrix representations for b-splines. In *Proceedings Pacific Graphics' 98. Sixth Pacific Conference on Computer Graphics and Applications (Cat. No. 98EX208)*, pages 37–43. IEEE, 1998. 3

[6] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, Zhaoyang Zeng, Hao Zhang, Feng Li, Jie Yang, Hongyang Li, Qing Jiang, and Lei Zhang. Grounded sam: Assembling open-world models for diverse visual tasks. *arXiv preprint*, 2024. 3