# 実映像ドライビングシミュレータにおける NeRF の活用と視野特性に基づいた高速化の検討

國信 綾斗 *1　張 ハンウェイ *2　栗 達 *1　川崎 洋 *2　小野 晋太郎 *1
福岡大学 *1　九州大学 *2

一般に、実映像において撮影した視点以外からの見えを解析的に再現することは原理的に不可能であり、シーンの三次元形状情報など何らかの知識や推定が必要である。このため、実映像によるドライビングシミュレータ (DS) の実現、すなわち自由な車線変更を含む走行シーンの再現は実用化が進んでいない。これに対し、深層機械学習に基づいて自由な視点からの見えを再現できる NeRF と呼ばれる技術が注目されている。しかし、NeRF によるレンダリング（描画）のプロセスは計算量が多く、DS として活用するためには十分なフレームレートの確保が課題となる。本研究では、NeRF の一種である F2-NeRF をベースとして、シーンの視野内の位置に応じて描画の品質を変更し、更にレンダリング処理を並列化することで処理時間を大幅に短縮し、平均 13.7 fps まで高速化できることを確認した。

# Utilizing NeRF in Real Image Driving Simulator and its Acceleration based on Field of View Characteristics

Ayato Kuninobu*1　Hanwei Zhang*2　Da Li*1　Hiroshi Kawasaki*2　Shintaro Ono*1
Fukuoka University*1　Kyusyu University*2

**Abstract** It is known that it is impossible to analytically reproduce a free-point view of real video, and some knowledge such as 3-D structure of the scene and/or estimation is required. For this reason, a driving simulator (DS) using real video, reproducing driving scenes with free lane changes, has not progressed in practical use. Regarding this issue, a technology called NeRF (Neural Radiance Field) that can reproduce free viewpoints based on deep learning is attracting attention. However, the rendering process using NeRF requires a large amount of calculation, and a problem arises in that the frame rate decreases when using it as a DS. In this study, we propose a method to change the rendering quality according to the position in the field of view of the scene, and to parallelize the rendering process based on F2-NeRF. The experimental result showed that the processing time was greatly reduced and the average frame rate reached 13.7 fps.

***Keyword:*** *Driving Simulator, Real Image, NeRF, Central/Peripheral Vision*

## 1.　はじめに

ドライビングシミュレータ (DS) は車両の走行状況を再現する仕組みとして道路交通や車両に関連する研究・開発に広く活用されており、自動運転技術の開発においてもその需要が高まっている。

DS における走行シーンの描画方法は、従来型の 3 次元 CG により実現する手法がほとんどであり、多種多様なシナリオを自由に再現することが可能であるが、CG モデルの構築にはコストを要する。これに対し、未だ実用化は非常に限定的であるものの、CG の代わりに撮影した実映像を用いる手法も知られており、CG モデルの構築処理を必要とすることなく、高い現実感により走行シーンを再現することが可能である。一般に、実映像において撮影した視点以外からの見えを解析的に再現することは原理的に不可能であり、シーンの三次元形状情報など何らかの知識や推定が必要である。

本研究では、NeRF[1) ] (Neural Radiance Field) と呼ばれる自由視点再現技術を用いて、視点を自由に変更可能な DS を実現することを考える。ここで課題となるのが NeRF によるレンダリング（描画処理）の計算時間であり、DS として用いるために十分なフレームレートを達成できないことが問題となる。そこで我々

は、自動車を運転する際に、ドライバの視野が中心視・有効視野・周辺視野に分かれることに着目し、視野内の位置に応じてレンダリングの品質を変えることで処理を軽量化し、さらに GPU による並列計算およびメモリ利用を最適化することで処理を高速化する手法を提案する。

## 2. 関連研究

NeRF とは、複数の視点からシーンを撮影しておき、視点（位置・姿勢）と、その際の見え方の関係を深層機械学習により学習させておくことで、別の視点（位置・姿勢）から見た画像を推定して再現する技術である。内部的には、ボリュームレンダリングと呼ばれる描画技法が使われる。すなわち、対象の空間を多数の格子に分割し、それらの色と密度を学習によって決定し、見え方を推定・再現（レンダリング、描画）する際は、各画素の視線方向ごとに色と密度を積算することで画素値を決定することができる。但し、これらの計算コストは一般に高く、一例として F2-NeRF[2] では、本稿 5. の実験条件において、解像度 960×450 ピクセルのシーン 1 枚をレンダリングするために約 3.9 s を要する。

この NeRF は、取り扱うシーンの種類によって大きく 2 つの手法に分けられる。1 つは、境界（奥行き方向の上限）があるシーンであり、極めて狭い範囲や特定の物体を対象とするものである。このようなシーンを扱う NeRF には、NeRF++[3]、instant-ngp[4]、mip-NeRF360[5] などがある。これらは、対象物を中心として周辺 360° 方向から内向きに撮影・学習させるのが基本であるため、車両の走行のように視点が広い範囲を動く状況には適していない。もう 1 つは、境界（奥行き方向の上限）のないシーンであり、ドローンや車両のように広範囲を移動し、視点が比較的自由に移動するケースである。このようなシーンを扱う NeRF には、F2-NeRF のほか、URF[6]、Block-NeRF[7] などがある。

自由視点の再現手法として、NeRF に続いて提案された別の新しいアプローチとして Gaussian Splatting[8] がある。Gaussian Splating (GS) は、対象シーン全体の形状が 3 次元ガウス分布の集合により構成されると仮定し、それを 2 次元に投影することで任意の視点を再現する。各分布のパラメータ（位置・向き、大きさ、透明度など）は深層機械学習により推定される。一般に GS は、NeRF と比較して、学習・レンダリングともに高速である。また、GS を動的なシーンに拡張したのが 4DGS[9] である。これにより他車両や歩行者などの移動物体が含まれるシーンも取り扱えるようになり、更にその GS を都市部の運転シーンに特化して展開した手法がが StreetGS[10] と DrivingGS[11] である。どちらも都市部の運転シーンの再現において高速な結果を示している。

このように GS と NeRF は旧来のラスタライズ方式とレイトレーシング方式の関係に似ている。そのため、リアルタイムレンダリングには GS が向いていると考えられるが、現実空間の視覚的な再現性という点においては、光を追跡して再現する NeRF の方が優れている可能性がある。

一方、ここまでに紹介した自由視点再現手法は、シーンをできるだけ忠実に再現することに主眼が置かれており、運転中のドライバの特性にまで着目してレンダリング処理を高速化することは検討されていない。本研究においては、ドライバの視野に応じてレンダリング処理を軽量化する手法を提案する。このような考え方は、様々なレンダリング手法に対して一般に適用できる可能性があるが、本研究では第一段階として、より基本的・歴史的な手法であり、高速化の余地が大きく、再現品質に優れる可能性のある NeRF を用いることとする。

## 3. 要素技術

まず、本研究で取り扱う要素技術である F2-NeRF、レイトレーシング、GPU による並列処理について簡潔に説明する。

### 3.1 F2-NeRF

F2-NeRF[2] とは、Wang らによって提案された NeRF の一種である。従来の NeRF では、入力となる画像群は、対象物体を中心に 360 ° 方向から取り囲むような視点（位置・姿勢）で撮影することを前提としてきた。これに対し F2-NeRF では、任意のカメラ視点に対応できるようになったことが特徴である。これにより自動車やドローンに積載したカメラや、その近傍からの自由な視点再現に適用できると考えられる。他にも、学習時間の短縮など、幾つかの特長も知られている。

具体的なレンダリング処理は一般的な NeRF と同様である。まず対象空間を格子に分割し、それらの色と密度を学習によって求めておく。レンダリング時は、指定する視点の位置・姿勢をもとに、画素数の分だけ対象空間に向かってレイ（Ray，後述）を飛ばし、空間を探索して色や密度を取得し、積算することで画素値を決定する。対象空間の格子は八分木構造により階層的に管理されており、レイが交差する格子を探索する部分（交差判定）に最も計算時間を要する。このレンダリング処理は、次に述べるレイトレーシングと同様である。

### 3.2 レイトレーシング

現実空間においては、光源から発射された光が空気中を通って直接、あるいは物体に反射を繰り返して目に届くことで色や明るさを持った像が得られる。レイトレーシングは、コンピュータグラフィクス (CG) におけるレンダリング手法の一種であり、先述の物理的な過程を逆方向に探索することで像（画素値の集合）を計算する。すなわち、視点からレイ[1]を飛ばして通過（交差）点や物体表面における反射等を模擬し、色や明るさを計算する方法をとる（図 1）。物理的な過程を模擬しているため、現実空間の光現象を可能な限り再現できることがメリットである。一方、画素数分だけレイ

[1] 光線の意味であるが、視点に光が届くのとは逆向きであるため、口語的に視線と言い換えることもある。

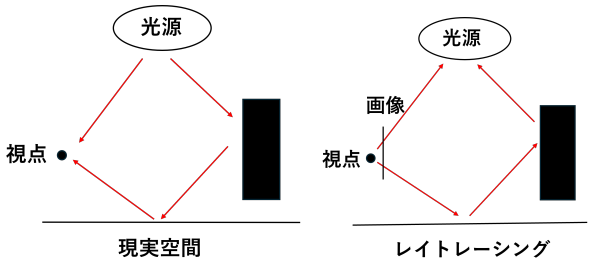



**Fig.**1 レイトレーシング処理の概要

を飛ばしてレイごとに処理を行う必要があるうえ、忠実に模擬するためには途中で更に複数のレイを飛ばす（反射方向が広がる場合など）ことになり、計算量が爆発的に増大するというデメリットもある。

### 3.3 GPU を利用した並列処理

GPU (Graphic Processing Unit) とは、コンピュータ内の演算装置の一種である。画像処理や CG 処理（大規模な積和演算、特に座標変換、頂点・ピクセル処理、陰影計算など）に特化した演算器で構成されており、近年では機械学習の演算にも広く活用されている。この GPU を用いた並列処理について簡潔に述べる（参考：NVIDIA[12]）。

GPU 上で実行する関数はカーネル関数と呼ばれ、カーネル関数の実行 1 回分に相当する処理をスレッドと呼ぶ。更に、スレッドの集まりはブロック、ブロックの集まりはグリッドと呼ばれる（図 2）。GPU 上での並列処理は、スレッド、ブロック、グリッドをそれぞれ 1〜3 次元で構成し、カーネル関数を並列に呼び出すことで実現される。

前述のレイトレーシング処理を例に、GPU を用いて並列化する仕組みを説明する。レイトレーシングは画素数分だけレイを飛ばし、それぞれのレイにおいて探索処理を行う。この処理はレイごとに独立しているため、レイの数だけスレッドを用意してカーネル関数を呼び出す（図 3）。コードで実装する際には、1 ブロックあたりのスレッド数とブロック数を指定し、その積がレイの数と同じになるようにすれば良い。

なお、カーネル関数は GPU 上のメモリ（ビデオメモリ、VRAM）上のデータしか扱うことができず、メインメモリは参照できない。従って、計算に必要なデータは事前に GPU メモリに転送しておく必要がある。GPU メモリは図 4 のように構成されており、グローバルメモリ、テクスチャメモリ、コンスタントメモリはどのスレッドからもアクセスできるが、テクスチャメモリとコンスタントメモリは読み込み専用である。共有メモリは同じブロック内で共有してデータのやり取りができるが、本研究では使用していない。

## 4. 提案手法：レンダリング処理の高速化

### 4.1 ドライバの視野特性に基づいた領域別処理

ヒトの視野のうち、文字や対象の形状をはっきり認識できる範囲（中心視）は 1〜2° 程度、中心視と同時に識別処理を行える範囲（有効視野）は ±4〜20° 程度

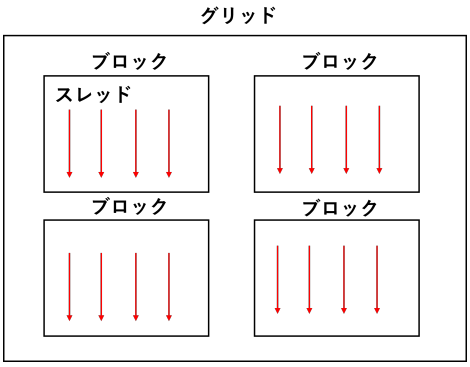



**Fig.**2 スレッド、ブロック、グリッドの関係

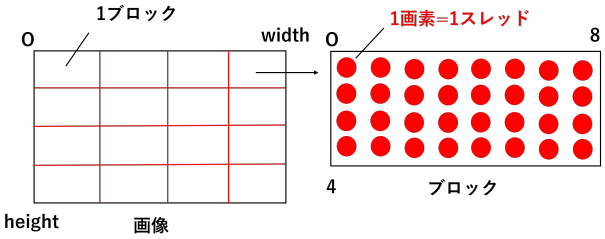



**Fig.**3 レイトレーシング処理の並列化

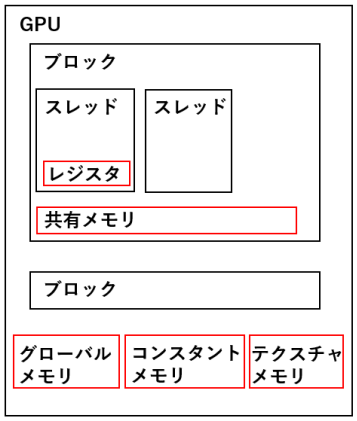



**Fig.**4 GPU メモリの構成

であり、走行速度や注視対象によっても変化することが知られている [13]。

この特性を考慮すると、シーン全体を必ずしも均一の品質で再現する必要はなく、視野内の範囲に応じて適度な品質で再現する手法が考えられる。範囲の定め方や各範囲に要求される再現性、ドライバの認知への影響などを検討するためには更に詳細な実験・分析が必要となるが、ここではまず、領域分割とレンダリングパラメータ変更により高速化が実現できることを実証する。

具体的には、レンダリングすべきシーン全体（幅 $W \times$ 高さ $H$）に対し、もっとも基本的な分割方法として中心領域（$w \times h$）と外側領域に分割する。いずれの領域も F2-NeRF によりレンダリング処理を行うが、レイ

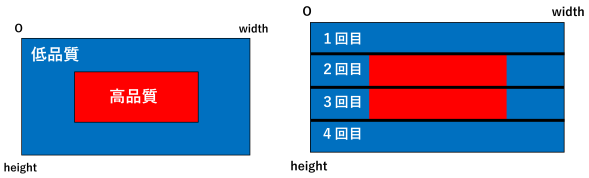



**Fig.**5 レンダリング品質の設定 (左) と分割処理 (右)

トレーシング処理の際に内部で行う空間の八分木探索の回数を、中心領域では最大 $N$ 回、外側領域では最大 $aN$ 回とする。

今回は試行的に $w = 0.5W,\ h = 0.5H,\ a = 0.5$ と設定する。これにより画像端から 1/4 の距離範囲、面積にして全体の 3/4 の範囲（図 5）においてレンダリングの探索回数を約半分とし、再現品質を抑えている。

### 4.2 GPU による並列処理

レンダリングにおいて必要となる視点の位置と、各画素に対応したレイの向きは、全スレッドにおいて共通に参照する必要のある情報である。よって、視点の位置をコンスタントメモリに、各画素に対応するレイの向きをテクスチャメモリに配置する。これらはいずれも読み込み専用の GPU メモリである。更に、GPU を利用した並列処理ではスレッド数を 32 の倍数で設定すると高速化できることが知られているため、スレッド数はそれに従って設定する。また、ブロックは 2 次元で構成する。

以上の内容について、元の F2-NeRF から変更して実装する。

### 4.3 分割処理による GPU メモリ消費量の軽減

木構造の探索において、結果を保持するためのメモリ消費量が爆発的に増えることが予想される。そのため、全画素数分のレイを同時に処理するのではなく分割数を指定して少しずつ処理するよう変更する（図 5）。分割数は領域別処理の範囲に応じて 4 に設定している。

## 5. 実験

提案した高速レンダリング手法を用いて走行シーンを運転者の視点で再現し、処理時間および再現性について性能を評価する。

走行シーンを 3 次元 CG 映像または実環境から映像として取得し、COLMAP[14, 15] により画像フレーム毎の視点（位置・姿勢）を計算する[2]。この視点情報と画像を入力として F2-NeRF を訓練する。

### 5.1 評価指標

処理時間は 2 つの指標により評価する。

- レイトレース時間 $T_R$： レンダリング処理の中核となる、レイを飛ばして交差箇所の探索と情報取得を行うのに要する時間。

[2]CG 環境ではシーンの構造や走行経路も内部的には 3 次元座標で表されているため、COLMAP により計算する必要はない。しかし、この CG は実映像を用いた処理を模擬する目的で導入しているため、それらの 3 次元情報は用いず、実映像と同様に処理する。

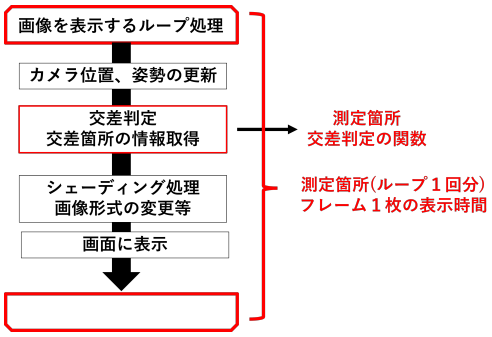



**Fig.**6 処理時間の評価指標

- フレーム時間 $T_F$： 表示すべき視点の位置・姿勢を与えてから画像フレーム 1 枚を生成し、更に表示するまでに要する時間。$1/T_F$ がフレームレートに相当。

これらの関係を図 6 に示す。

再現性は、提案手法により生成した画像と、真値となる画像の差異 (PSNR) により評価する。NeRF を訓練するために与えた画像を真値とし、その視点からの見えを提案手法により再現する。それ以外の視点については一般に真値が不明であるため、今回は評価対象としていない。

### 5.2 実験条件

走行シーンは、CG 映像および実映像のデータセットから取得する。訓練画像と同じ視点からの見えを生成する処理を 5 回実行し、$T_F, T_R$ は 1 回・1 フレームあたりの平均として求める。GPU には NVIDIA GeForce RTX3090 を用いる。

#### 5.2.1 CG 映像データ

CG 映像データは、レーシングゲーム「グランツーリスモ 7」[3] の首都高速道路コースから取得する。2 車線分の幅に対して概ね均等に 6 つの視点を設定し、各視点が約 60km/h で走行し、30 fps でキャプチャする。なお、ゲームを手動で操作してキャプチャするため、視点と速度は多少の誤差を含んでいる。

キャプチャ画像のうち、15 フレーム毎（2 fps 相当）に切り出した 389 枚の画像を F2-NeRF の訓練用に用いる。画像解像度は、訓練・レンダリングともに 960×450 である。

#### 5.2.2 実映像データ

実映像データは、日本の市街地を撮影した CoVLA[16] の「2022-07-21–15-11-45–44_first」に含まれる 600 枚の画像を学習に使用する。

処理の都合上、画像解像度は 32 の倍数である必要があるため、960×576 に調整している。枚数と解像度以外は、CG データセットと同様の条件である。

### 5.3 比較ケース

比較評価を行うため、以下の 4 つのケースを設定する。

[3]Gran Turismo®7. ©2022 Sony Interactive Entertainment Inc. Developed by Polyphony Digital Inc.

1. 提案手法（領域・並列・分割）: 領域別処理、GPU による並列処理、分割処理を行う場合
2. 提案手法（領域・並列）: 領域別処理および GPU による並列処理を行う場合
3. 提案手法（並列）: GPU による並列処理のみを行う場合
4. オリジナル手法： F2-NeRF から変更しない場合

1, 2, 3 は第 4. 章に説明した手法であり、4 では Github[4] 上で公開されているコードに実装されている関数のみを用いる。

## 6. 結果

### 6.1 CG 映像データ

CG データによる評価結果を表 1、レンダリング結果を図 7 に示す。提案手法（領域別・並列処理）により、レイトレース時間 $T_R$ は約 99%短縮、フレーム時間 $T_F$ は 98%短縮と、劇的に高速化できていることが分かる。これは、交差判定が多くなると予想される建物や道路が画像の端の方に多数位置しており、探索処理の上限を下げた恩恵が広範囲にわたって得られたためと考えられる。なお、並列処理による時間短縮効果は 15%程度である。

画像の再現品質に関しては、PSNR はオリジナル手法に対し 0.56 dB の低下であり、劣化は軽微であると言える。実際のレンダリング結果（図 7）においても、外側部分の劣化は目立たなくなっている。

### 6.2 実映像データ

実映像データによる評価結果を表 2、レンダリング結果を図 8 に示す。提案手法（領域別・並列処理）により、レイトレース時間 $T_R$ は約 74%短縮、フレーム時間 $T_F$ は 78%短縮されている。しかし、提案手法（並列処理のみ）の場合、$T_R$ は短縮されたものの $T_F$ は増加しており、交差判定処理よりも後の部分において時間がかかっていると考えられる。また、提案手法（領域別・並列・分割処理）において PSNR はオリジナル手法よりも 0.54 dB 改善している。これは、オリジナル手法においてレンダリング処理の一部が追い着かないまま後続の処理を行っている可能性があり、これが処理の高速化により改善されたためと推測される。さらに、提案手法（領域別・並列処理）に比べてフレームレートが約 6 倍大きくなっている。これは画角内に写る建物が多く、全体的に探索処理の回数が増加したことによるメモリ消費の負荷が分割によって軽減されたためと考えられる。

## 7. おわりに

本論文では、NeRF による走行シーンの再現において、画像の一部領域でレイトレーシングの探索回数を抑え、GPU による並列化と併用することで、学習視点での画像劣化を軽微に抑えつつ計算時間を大幅に削減し、高速化できることを示した。今後の課題として、自由視点（学習外の視点）での評価、走行速度や視線に応じた動的な領域分割、シーン特性に応じた適切なレンダリングパラメータの設定などが挙げられる。

[4]https://github.com/Totoro97/f2-nerf（2024/11/2 参照）


## 謝辞

本研究の一部は JSPS 科研費 JP21K03962 および NEDO SIP（スマートモビリティプラットフォームの構築）の支援を受けた。


## 参 考 文 献

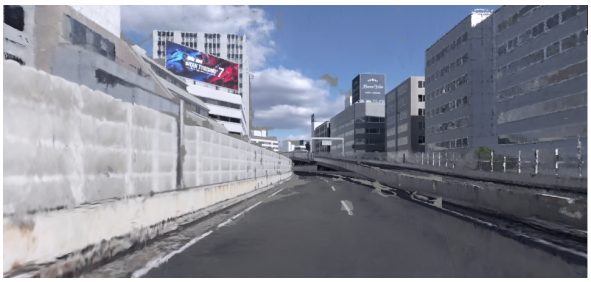

1) Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: representing scenes as neural radiance fields for view synthesis. *Commun. ACM*, Vol. 65, No. 1, p. 99–106, December 2021.
2) Peng Wang, Yuan Liu, Zhaoxi Chen, Lingjie Liu, Ziwei Liu, Taku Komura, Christian Theobalt, and Wenping Wang. F2-nerf: Fast neural radiance field training with free camera trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4150–4159, June 2023.
3) Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields, 2020.
4) Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, Vol. 41, No. 4, July 2022.
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)*, pp. 5470–5479, June 2022.
6) Konstantinos Rematas, Andrew Liu, Pratul P. Srinivasan, Jonathan T. Barron, Andrea Tagliasacchi, Tom Funkhouser, and Vittorio Ferrari. Urban radiance fields. *CVPR*, 2022.
7) Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P. Srinivasan, Jonathan T. Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8248–8258, June 2022.
8) Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, Vol. 42, No. 4, July 2023.
9) Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20310–20320, June 2024.
10) Yunzhi Yan, Haotong Lin, Chenxu Zhou, Weijie Wang, Haiyang Sun, Kun Zhan, Xianpeng Lang, Xiaowei Zhou, and Sida Peng. Street gaussians: Modeling dynamic urban scenes with gaussian splatting. In *ECCV*, 2024.
11) Xiaoyu Zhou, Zhiwei Lin, Xiaojun Shan, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. Drivinggaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21634–21643, June 2024.
12) NVIDIA. Cuda c++ programming guide. `https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html`. (参照 2024 年 11 月 2 日).
13) 三浦利章. 視覚的注意と安全性. 照明学会誌, Vol. 82, No. 3, pp. 180–184, 1998.
14) Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
15) 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.
16) Hidehisa Arai, Keita Miwa, Kento Sasaki, Yu Yamaguchi, Kohei Watanabe, Shunsuke Aoki, and Issei Yamamoto. Covla: Comprehensive vision-language-action dataset for autonomous driving, 2024.

**Table** 1 CG 映像データ（画像サイズ 960×540）による評価結果

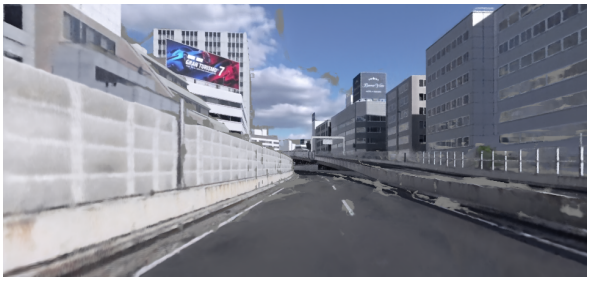
| | レイトレース時間 $T_R$ [ms] | フレーム時間 $T_F$ [ms] | フレームレート $1000/T_F$ [fps] | 画像品質 PSNR [dB] |
|---|---|---|---|---|
| 提案手法（領域・並列・分割） | 40.91 | 73.48 | 13.61 | 23.24 |
| 提案手法（領域・並列） | 37.30 | 78.39 | 12.75 | 23.24 |
| 提案手法（並列） | 3126.83 | 3362.18 | 0.30 | 23.80 |
| オリジナル手法 | 3710.06 | 3941.97 | 0.25 | 23.80 |

**Table** 2 実映像データ（画像サイズ 960×576）による評価結果

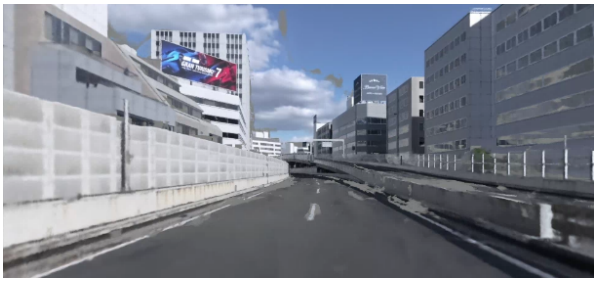
| | レイトレース時間 $T_R$ [ms] | フレーム時間 $T_F$ [ms] | フレームレート $1000/T_F$ [fps] | 画像品質 PSNR [dB] |
|---|---|---|---|---|
| 提案手法（領域・並列・分割） | 35.11 | 72.77 | 13.74 | 28.53 |
| 提案手法（領域・並列） | 264.81 | 447.37 | 2.24 | 28.28 |
| 提案手法（並列） | 970.66 | 3460.18 | 0.29 | 27.99 |
| オリジナル手法 | 1003.31 | 2018.75 | 0.50 | 27.99 |



(a) 領域別処理を行った場合（外側領域が低品質）



(b) 参考：領域別処理を行わない場合（全体が均一品質）



(c) 視点を 2 車線の中央に移動した場合（学習外視点）

**Fig.7** CG 映像データによるレンダリング結果

(a) 領域別処理を行った場合（外側領域が低品質）

(b) 参考：領域別処理を行わない場合（全体が均一品質）

(c) 視点を右に平行移動した場合（学習外視点）

**Fig.8** 実映像データによるレンダリング結果