An Adaptive Shooting and Bouncing Ray Method Based on Q-Learning for Efficient Synthetic Aperture Radar Imaging Simulation

Dayong Tian , Shuo Wang , Md. Gazi Salahuddin , Xiaoyang Li *

Posted Date: 15 June 2026

doi: 10.20944/preprints202606.1117.v1

Keywords: synthetic aperture radar; SAR imaging; Q-learning; adaptive SBR; ray-tracing; electromagnetic simulation

 doi:10.20944/preprints202606.1117.v1

Article

An Adaptive Shooting and Bouncing Ray Method Based on Q-Learning for Efficient Synthetic Aperture Radar Imaging Simulation

Dayong Tian, Shuo Wang, Md. Gazi Salahuddin and Xiaoyang Li *

School of Electronics and Information, Northwestern Polytechnical University, Xi'an, China
* Correspondence: lixiaoyang@nwpu.edu.cn

Abstract

Fast synthetic aperture radar (SAR) imaging simulation is required by many computer vision applications. Although the shooting and bouncing ray (SBR) method has significantly accelerated electric field calculation, the number of ray tubes is still the bottleneck for SAR image simulation speed. This letter proposes an innovative adaptive SBR method driven by Q-learning for accelerated SAR imaging simulation. The core strategy is to convert the ray tube allocation into a reinforcement learning problem. The ray-shooting plane is dynamically partitioned into localized patches, where a Q-learning agent intelligently scales the ray density in real-time. By observing the geometric features of the target surface, the agent learns to employ coarser ray tubes in flat regions to eliminate redundant computation, while deploying denser ray tubes in complex areas. A multi-objective reward function is designed to balance accuracy against computational resource consumption. Numerical experiments demonstrate that the proposed Q-learning-based SBR method drastically reduces computational cost while preserving imaging similarity.

Keywords: synthetic aperture radar; SAR imaging; Q-learning; adaptive SBR; ray-tracing; electromagnetic simulation

1. Introduction

Computer vision for synthetic aperture radar (SAR) images requires fast SAR imaging simulation. For example, adversarial attacks for object detection [1] models on SAR images require fast SAR imaging simulation to change the shapes of targets, spatial layout of corner reflectors or dynamically relocating maritime vessels, to generate a large number of candidate examples for searching. Data augmentation for SAR image object detection [2] or false target recognition [3] also requires fast SAR imaging simulation, because traditional data augmentation for optical images, such as rotations, scaling and color jittering, are physically meaningless for SAR images.

There are two main trends in accelerating SAR imaging simulation. One trend is using deep neural networks, e.g., generative adversarial networks to generate SAR images [4]. However, these data-driven methods are not physics-based and therefore they cannot be used for applications that require accurate physical modeling. Another trend is using ray-tracing-based electromagnetic simulation methods, e.g., Shooting and Bouncing Ray (SBR) method [5], to accelerate the electric field calculation in SAR imaging simulation. Although SBR-based electromagnetic calculation methods have significantly increased the speed of electric field calculation, they still need to be improved for the above-mentioned applications.

SBR-based SAR imaging simulation generally consists of two stages. Firstly, it shoots a large number of ray tubes from the radar to the target to compute the electric field from each intersection point or scattering center. Then, it aggregates the linear frequency modulation (LFM) signals from all these points to obtain the raw data for subsequent range and azimuth compression. Since SBR computations are mutually independent across different ray tubes, the ray tracing and intersection

 doi:10.20944/preprints202606.1117.v1

testing phases can be efficiently accelerated by NVIDIA OptiX which utilizes RT Cores for hardware-level acceleration. However, the subsequent summarization of the LFM signals from every individual scatter center or intersection point presents a severe non-parallelizable bottleneck. Aggregating a massive number of these signals is highly time-consuming (Figure 1). The motivation of the proposed method is to reduce the number of simulated rays, thereby fundamentally reducing the number of LFM signals to be aggregated.

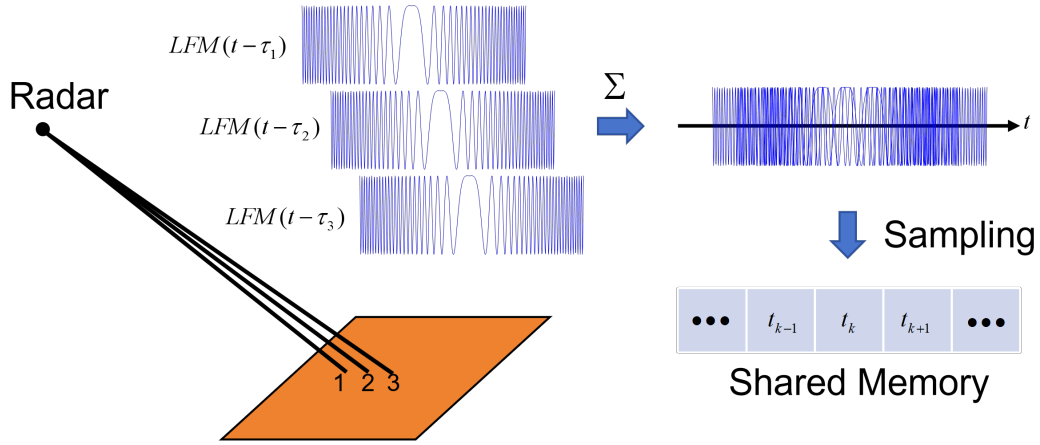


Figure 1. Illustration of the computational bottleneck in SBR-based SAR imaging simulation. Three ray tubes are shot from the radar to the target, where each scattering center returns an LFM signal with a specific propagation delay τ_i $(i = 1, 2, 3)$. To obtain the raw data, all LFM signals must be aggregated at each discrete sampling time t_k in the time domain, i.e., $\sum_i LFM(t_i)$ where $t_i = t_k + \tau_i$. This aggregation is strictly sequential on CPUs. On GPUs, parallel aggregation still suffers from a severe performance bottleneck due to atomic contention and memory write conflicts when processing a massive number of signals.

Numerous studies have proposed methods to reduce the number of rays in electromagnetic simulation for radar cross section (RCS) calculation as well as in computer graphics for rendering. However, none of these ray reduction methods are directly suitable for SAR imaging simulation. When calculating RCS, the far fields from all scattering centers are summarized at a single observation point. For a simple rectangular or triangle facet, we can use different ray tube layouts to calculate RCS without losing accuracy (Figure 2). In rendering, Monte Carlo estimators are generally used to calculate the integrals in the rendering equation. The phase shifts caused by spatial propagation delays are not explicitly modeled in rendering, and therefore the ray reduction methods do not need to consider the phase preservation issue. Nevertheless, the phase issue is critical for SAR imaging and therefore we need to design a specific ray reduction method for SBR-based SAR imaging simulation.

As shown in Figure 2, for a simple rectangular perfectly electric conducting (PEC) target, we can merge the dense grid into sparser grids or even a single ray tube to compute RCS. However, for SAR imaging, the merging depends on the azimuth and range resolution, receiver position, etc.

To address these challenges, this letter introduces an innovative intelligent SBR framework powered by reinforcement learning to accelerate SAR imaging simulation. We formulate the ray tube scaling problem as a sequential decision-making process solved via the Q-learning algorithm. The ray-shooting plane is segmented into localized patches, each governed by a trained Q-learning agent. By analyzing a concise set of localized geometric features, the agent adaptively dictates the optimal layout of the ray tubes. It deploys coarser ray tubes in smooth, flat domains to filter out computational redundancy, while triggering fine-grained ray tube splitting in complex geometric regions to preserve phase and amplitude accuracy. Guided by a multi-objective reward function that explicitly balances echo precision against ray-tracing cost, the framework learns the optimal sampling density autonomously.

 doi:10.20944/preprints202606.1117.v1

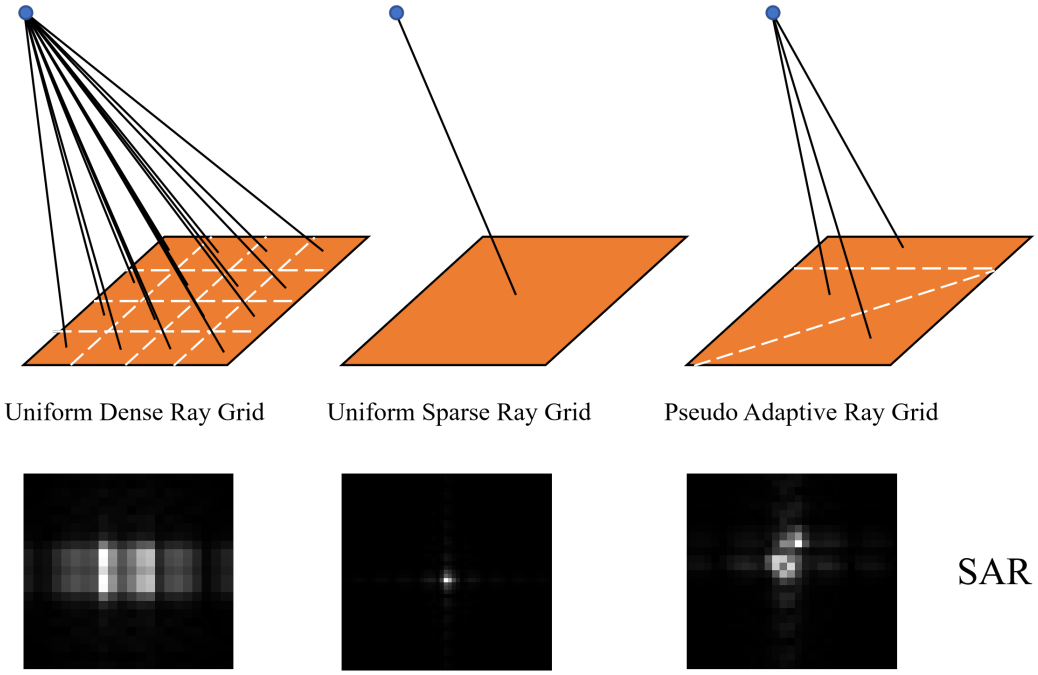


Figure 2. Illustration of the phase preservation requirement in SBR-based SAR imaging simulation. In all three cases, the RCS can be correctly calculated based on the far-field expression of the time-domain physical optics (TDPO) integral. However, they have different SAR imaging results due to the different phase shifts of different ray tube layouts. Only the uniform dense grid layout can obtain the correct imaging result. Our proposed method learns to intelligently allocate ray tubes to preserve phase shifts while reducing ray numbers.

2. Related Works

Various ray reduction methods have been developed in RCS calculation and rendering. In RCS calculation, one must calculate the integrals on the intersection area between ray tubes and target facets. In rendering, one needs to estimate the integrals by each intersection point between a ray and a target facet. Therefore, their ray reduction methods are different.

2.1. Ray Reduction in RCS Calculation

Suk *et al.* [6] proposed a multi-resolution grid algorithm to reduce ray counts in radar cross section (RCS) calculation. Jin *et al.* [7] and Bang *et al.* [8] employed space-division algorithms for ray reduction. Huo *et al.* [9] proposed tracing individual rays and introducing virtual tubes, thereby reducing the number of traced ray tubes. Zhao *et al.* [10] employed a facet-neighborhood search method to reduce the number of ray-facet intersection tests. Hu *et al.* [11] proposed an adaptive ray-tube splitting method based on whether the tube's lateral edges intersect the target facets. They further imposed a radius threshold to prevent excessive ray-tube splitting on small facets.

2.2. Ray Reduction in Rendering

Estimating the integrals in the rendering equation [12] using ray tracing methods is a fundamental problem in computer graphics. As only a limited number of rays can be traced, the estimations suffer from high variance, which manifests as noise in the rendered images. Researchers have developed various methods to reduce the variance, including sampling, denoising, super-resolution, etc. Among them, sampling is the most relevant to our work. Sampling methods can be categorized into two types: importance sampling and adaptive sampling [13]. Importance sampling methods try to estimate a probability distribution function (PDF) that can align with the product of the measurement contribution function and the pixel reconstruction filter. Adaptive sampling methods start from a low-sample MC rendering input and then strategically allocate additional samples to enhance the image quality.

 doi:10.20944/preprints202606.1117.v1

2.2.1. Importance Sampling

Importance sampling improves Monte Carlo (MC) integration by drawing samples from a proposal distribution that closely approximates the integrand, thereby reducing estimator variance for a fixed sample budget. Early neural approaches to this problem cast path guiding as a reinforcement learning (RL) task. Dahm and Keller [14] formulated the selection of light-transport directions as an RL problem in which an agent progressively learns the spatial radiance distribution, showing that the resulting policy substantially shortens average path lengths and reduces wasted evaluations—a principle directly analogous to our goal of eliminating redundant ray tubes in flat target regions. Zheng and Zwicker [15] trained a normalizing-flow network to warp the primary sample space so that drawn samples concentrate on high-radiance directions; the learned non-linear mapping adapts online and converges faster than histogram-based guides.

Moving beyond single-bounce guidance, Bako *et al.* [16] proposed an offline, scene-independent deep importance sampler (ODIS) that reconstructs the full incident radiance at a surface point from a sparse set of neighbours, enabling effective first-bounce guidance with as few as one sample per pixel. Müller *et al.* [17] introduced Neural Importance Sampling, which frames the density learning problem as minimising the KL divergence between a normalising-flow proposal and the target distribution and extends naturally to joint multi-dimensional path prefixes in primary sample space. Their follow-up, Neural Control Variates [18], augments any MC estimator with a learned surrogate integrand whose known integral is subtracted from the estimate, achieving unbiased variance reduction without modifying the underlying sampler.

Deep-learning path guiding matured substantially in 2021. Zhu *et al.* [19] demonstrated that photon traces emitted from light sources carry rich directional information; feeding these sparse photons into an offline-trained encoder-decoder yields high-fidelity sampling distributions for arbitrary scene regions. A hierarchical extension [20] combined path samples with photon data in a quad-tree representation, enabling finer-grained directional resolution at interactive update rates. Dong *et al.* [21] compressed spatial-directional distributions into a compact implicit neural representation (Neural Parametric Mixtures), decoding them on demand into parameter-mixing models that support fast, closed-form importance sampling.

The most recent path-guiding work targets the constraints of real-time budgets. Huang *et al.* [22] proposed an online framework built around the Normalized Anisotropic Spherical Gaussian (NASG) mixture model: a small MLP trained on stochastic ray samples learns the full light-transport product distribution and can be sampled in closed form on the GPU, removing the warm-up phase required by prior methods. Lu *et al.* [23] introduced Voxel Path Guiding (VXPG), which stores irradiance and geometry in boundary-aware voxels shared across shading points and uses a streamlined K-means clustering step to group voxels by visibility and irradiance similarity; unbiased sampling within the selected voxel then achieves state-of-the-art perceptual error under an equal-time budget.

2.2.2. Adaptive Sampling

While importance sampling redistributes the probability mass used to generate each sample, adaptive sampling addresses a complementary question: how many samples should be allocated to each spatial region? Dachsbacher [24] pioneered the use of machine learning for ray-budget decisions by classifying scene regions according to their visibility configuration and applying perception-driven level-of-detail control; co-occurrence matrices over triangle clusters provided the feature representation. The approach established that learned spatial classifiers can replace hand-crafted variance metrics without loss of image quality.

Kuznetsov *et al.* [25] proposed DASR, the first end-to-end convolutional adaptive sampler for MC rendering. A CNN ingests a one-sample-per-pixel noisy image together with auxiliary geometry buffers and outputs a per-pixel sampling map that redistributes an additional three spp budget toward high-error regions, outperforming classical variance-based heuristics across diverse scene types. Vogels *et al.* [26] extended this idea with an iterative two-pass scheme: after an initial fixed-rate

 doi:10.20944/preprints202606.1117.v1

render, an error-prediction network forecasts per-pixel error after denoising, and subsequent passes double the total sample count by concentrating new samples at predicted error peaks.

Huo *et al.* [27] recast the per-pixel sample-count decision as a deep RL problem, training a Deep Q-Network (DQN) on offline datasets to guide where to evaluate the incident radiance field. A companion reconstruction network then synthesises the final four-dimensional radiance field from the adaptively acquired samples. The DQN formulation is the most direct precedent for our work: like our Q-learning agent, it observes scene features, takes discrete resource-allocation actions, and receives rewards tied to both accuracy and cost—albeit in the optical rather than the SAR domain.

Hasselgren *et al.* [28] extended adaptive sampling into the temporal domain, incorporating motion vectors and previous denoised frames as inputs to the sample predictor so that spatio-temporal coherence reduces both per-frame cost and flickering artefacts. Salehi *et al.* [29] moved away from empirical noise estimates by fitting closed-form analytic noise distributions to per-pixel rendering statistics; sampling maps are then derived from these distributions without requiring stored sample cascades, yielding faster training and higher quality at elevated sample counts. Most recently, Firmino *et al.* [30] showed that denoiser-output variance computed via first-order Taylor expansion and automatic differentiation is a reliable guide for iterative sample redistribution, achieving superior equal-time error across diverse scenes without training any additional network.

3. Methodology

As the proposed method incorporates methods in signal processing (Range Doppler), electromagnetic simulation (SBR) and reinforcement learning (Q-learning), the notations may be confusing for readers from different backgrounds. For example, s is usually used to represent signal in signal processing, but it is also used to represent state in reinforcement learning. λ is usually used to represent wavelength in electromagnetics, but it is also used to represent weight decay factor in reinforcement learning. Therefore, we divided our method into three subsections. Within each subsection, we use the notations commonly used in the corresponding field and clarify them separately. Some widely-used cross-disciplinary notations and their meanings are listed in Table 1.

Table 1. Cross-disciplinary Notations and Confounding Meanings.

Notations	**Signal Processing** (3.1)	**Electromagnetics** (3.2)	**Reinforcement Learning** (3.3)
s	Transmitted/received signal	Target surface area / Scattering mechanism (as subscript, e.g., k_s)	Environment state
λ	Operational wavelength	Operational wavelength	Weight decay factor
η	Slow-time variable / Additive white noise	Intrinsic impedance of media	Learning rate
k	Discrete sampling index / Chirp rate	Wavenumber in free space	Iteration index / Step counter
μ	Statistical mean / Expectation	Magnetic permeability	Statistical mean of distribution
σ	Radar cross section (RCS) / Standard deviation	Electrical conductivity	Standard deviation of policy/noise
ϵ	Residual error / Precision threshold	Complex permittivity	Convergence threshold
j	Imaginary unit	Imaginary unit	Grid cell index / Spatial counter
i	Imaginary unit	Imaginary unit / Incident wave (as subscript, e.g., E_i)	Grid cell index / Spatial counter

 doi:10.20944/preprints202606.1117.v1

3.1. Range Doppler Method for SAR Imaging Simulation

Range Doppler method is a widely used algorithm for broadside SAR imaging. Suppose the radar emits a linear frequency modulation (LFM) signal, the echo signal can be expressed as

$$s(t,\tau_i) = \sum_{i=1}^{N} E_{\mathbf{r}_i} \exp\left\{ j2\pi \left[f_c(t-\tau_i) + \frac{B}{2T}(t-\tau_i)^2 \right] \right\}, \tag{1}$$

where N is the number of scattering centers, $E_{\mathbf{r}_i}$ is the electric field received from the i-th scattering center, $\mathbf{r}_i$ is the vector from the i-th scattering center to the receiver, f_c is the carrier frequency, B is the bandwidth, T is the pulse duration, and τ_i is the time delay corresponding to the two-way travel time from the radar to the i-th scattering center. The Range-Doppler method applies azimuth and range matching filters in the slow-time and fast-time domains, respectively, to obtain a SAR image. Each time the radar emits an LFM pulse, the range filter is applied to the echo signal to obtain the range profile. The range filter is given by:

$$h_r(t) = \exp\left\{ -j2\pi \left[f_c t + \frac{B}{2T} t^2 \right] \right\}. \tag{2}$$

All echo signals received at different radar positions are processed by the range filter to obtain a range history. Then, the azimuth filter is applied to the range history to obtain the SAR image. The azimuth filter is given by:

$$h_a(t) = \exp\left\{ -j2\pi \frac{2v^2 t^2}{\lambda r_i} \right\}, \tag{3}$$

where v is the platform velocity, λ is the wavelength, and r_i is the range from the radar to the i-th scattering center. In real applications, we do not know the exact position of i-th scattering center, so a reference range r_0 is used in azimuth matching filter. The azimuth and range resolution are $\Delta_a = L/2$ and $\Delta_r = c/(2B)$, respectively, where L is the antenna length and c is the speed of light. To preserve the aspect ratio of targets, azimuth resolution is generally set to be equal to the range resolution, i.e. $L = c/B$.

In SAR simulation, we usually suppose a targeting imaging area, e.g., a square area at the center of XOY plane. According to the flight trajectory, we can calculate the minimum and maximum ranges from the radar to the imaging area, i.e., $r_{\min}$ and $r_{\max}$. Then, we can determine the number of range bins by $N_r = (r_{\max} - r_{\min})/\Delta_r * f_s/B$, where f_s is the sampling frequency. Each time the radar transmits an LFM signal, the sampled echo will be stored as a column of a signal matrix $\mathbf{S}$. The range matching filter is applied to each column of $\mathbf{S}$ and then the azimuth matching filter is applied to each row of the range-filtered signal matrix. Finally, we calculate the dB value of the azimuth and range filtered signal matrix to obtain the SAR image. The dynamic range of the SAR image is determined by the maximum and minimum values in the azimuth and range filtered signal matrix, which is usually set to 30 dB or 40 dB for visualization.

3.2. SBR Method for Electric Field Calculation

3.2.1. First-Order scattering

The first illumination on the target surface is computed by closed-form TDPO integration on triangular facets. Every facet can be treated independently and electromagnetic echo calculation can be performed in parallel.

For a time-varying plane wave, the incident wave can be expressed as:

$$\mathbf{E}_i(\mathbf{r},t) = \mathbf{E}_i(\mathbf{r}) \cdot E_0(t), \tag{4}$$

where $\mathbf{E}_i(\mathbf{r})$ indicates the amplitude and direction of the incident electric field at position $\mathbf{r}$. $E_0(t)$ indicates the wave form in time domain. As SAR imaging simulation is performed in the far-field region, the incident wave can be approximated as a plane wave. The far-field expression of TDPO

 doi:10.20944/preprints202606.1117.v1

integral for electromagnetic scattering from a PEC surface illuminated by a time-varying plane wave is given as:

$$\mathbf{E}_s(\mathbf{r}_i, t) = \frac{1}{2\pi r_i c}\hat{\mathbf{k}}_s \times \hat{\mathbf{k}}_s \times [\hat{\mathbf{n}}' \times (\hat{\mathbf{k}}_i \times \hat{\mathbf{e}}_i)] \int_{S_i} \frac{\partial}{\partial t'} E_i(t')ds \tag{5}$$

where r_i is the distance from the scattering center to the receiver, $\hat{\mathbf{k}}_s$ and $\hat{\mathbf{k}}_i$ are the unit propagation vectors of the scattered and incident waves, respectively, $\hat{\mathbf{e}}_i$ is the polarization vector of the incident wave, and $\hat{\mathbf{n}}'$ is the unit normal vector of the scattering surface. $E_i(t')$ represents the incident electric field waveform at the retarded time t', defined as $t' = t - \tau_i - r_i/c - (\hat{\mathbf{k}}_i - \hat{\mathbf{k}}_s) \cdot \mathbf{r}'/c$, where τ_i is the reference time delay for the i-th scattering center, $\mathbf{r}'$ is the position vector from the scattering center to the surface element, and S_i is the effective illumination area of the i-th scattering center.

Accordingly, the surface integral $I(t) = \int_{S_i} \frac{\partial}{\partial t'} E_i(t')ds$ can be expressed as:

$$I(t) = \frac{c}{\alpha^2} E_0(t) * \sum_{m=1}^{N_{edge}} I_m(t), \tag{6}$$

where $E_0(t)$ is the wave form in time domain and $I_m(t)$ indicates the contribution of the m-th edge of S_i. $I_m(t)$ can be expressed as:

$$I_m(t) = \boldsymbol{\alpha}^* \cdot \Delta\mathbf{e}_m \frac{c}{(\hat{\mathbf{k}}_i - \hat{\mathbf{k}}_s) \cdot \Delta\mathbf{e}_m} \left[\epsilon(t - t'_m) - \epsilon(t - t'_{m+1})\right], \tag{7}$$

in which $\boldsymbol{\alpha}$ is the projection of $\hat{\mathbf{k}}_i - \hat{\mathbf{k}}_s$ on surface S_i, $\boldsymbol{\alpha}^*$ is obtained by rotating $\boldsymbol{\alpha}$ by 90 degrees clockwise on S_i, $\Delta\mathbf{e}_m$ is the vector of the m-th edge, and ϵ is the Heaviside step function. Eq. (7) only works when $\alpha \neq 0$. When $\alpha = 0$, $I(t)$ can be expressed as:

$$I(t) = \Delta S \frac{\partial}{\partial t} E_0(t - \tau_i - r_i/c - (\hat{\mathbf{k}}_i - \hat{\mathbf{k}}_s) \cdot \mathbf{r}_{\mathbf{o}'}/c), \tag{8}$$

where $\mathbf{r}_{\mathbf{o}'}$ is the position vector from the scattering center to the center of S_i, and ΔS is the area of S_i.

3.2.2. Multiple Bounces Scattering

In SBR, each ray is traced through multiple bounces obeying Snell's law as shown in Figure 3. The N-th bounce of incident electric field can be expressed as:

$$\mathbf{E}_i^{(N)}(\mathbf{r}_i, t) = \mathbf{E}_i^{(N)}(\mathbf{r}) \cdot E_0(t). \tag{9}$$

The wave direction the N-th-order incident ray is $\hat{\mathbf{k}}_i^{(N)}$. According to the Snell's reflection law, the direction of the N-th-order reflected ray is:

$$\hat{\mathbf{k}}_r^{(N)} = \hat{\mathbf{k}}_i^{(N)} - 2(\hat{\mathbf{k}}_i^{(N)} \cdot \hat{\mathbf{n}})\hat{\mathbf{n}}. \tag{10}$$

The N-th-order reflection direction is the (N+1)-th-order incident direction, i.e., $\hat{\mathbf{k}}_i^{(N+1)} = \hat{\mathbf{k}}_r^{(N)}$. $E_i^{(N)}(\mathbf{r})$ can be locally decomposed into its horizontal and vertical polarization components, i.e.,

$$\mathbf{E}_i^{(N)}(\mathbf{r}) = E_{i,h}^{(N)}\hat{\mathbf{h}}_i^{(N)} + E_{i,v}^{(N)}\hat{\mathbf{v}}_i^{(N)}, \tag{11}$$

where $\hat{\mathbf{h}}_i^{(N)}$ and $\hat{\mathbf{v}}_i^{(N)}$ are the unit vectors of the horizontal and vertical polarization components, respectively. They can be expressed as:

$$\begin{cases} \hat{\mathbf{h}}_i^{(N)} = \dfrac{\hat{\mathbf{k}}_i^{(N)} \times \hat{\mathbf{n}}}{\|\hat{\mathbf{k}}_i^{(N)} \times \hat{\mathbf{n}}\|} \\ \hat{\mathbf{v}}_i^{(N)} = \hat{\mathbf{h}}_i^{(N)} \times \hat{\mathbf{k}}_i^{(N)} \end{cases} \tag{12}$$

 doi:10.20944/preprints202606.1117.v1

The polarization vectors of the reflected ray are given by:

$$\begin{cases} \hat{\mathbf{h}}_r^{(N)} = \hat{\mathbf{h}}_i^{(N)} \\ \hat{\mathbf{v}}_r^{(N)} = -\hat{\mathbf{h}}_r^{(N)} \times \hat{\mathbf{k}}_r^{(N)} \end{cases} \tag{13}$$

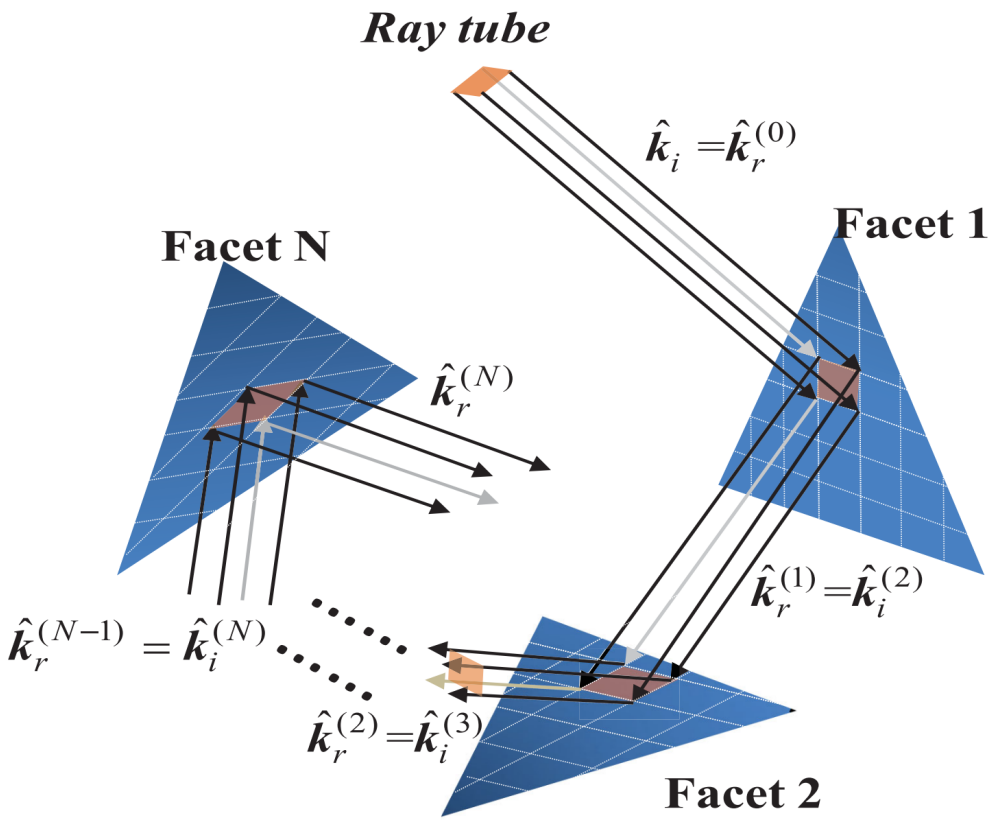


Figure 3. Illustration of multiple bounces in SBR.

The reflected electric field of the N-th bounce can be expressed as:

$$\mathbf{E}_r^{(N)}(\mathbf{r}) = E_{i,h}^{(N)} R_{hh} \hat{\mathbf{h}}_r^{(N)} + E_{i,v}^{(N)} R_{vv} \hat{\mathbf{v}}_r^{(N)}, \tag{14}$$

in which R_{hh} and R_{vv} are the Fresnel reflection coefficients for the horizontal and vertical polarization components, respectively. For PEC targets, $R_{hh} = -1$ and $R_{vv} = 1$.

For the first bounce, $\hat{\mathbf{k}}_i^{(1)} = \hat{\mathbf{k}}_i$ and $\mathbf{E}_i^{(1)}(\mathbf{r}) = \mathbf{E}_i(\mathbf{r})$. For the (N+1)-th bounce, $\hat{\mathbf{k}}_i^{(N+1)} = \hat{\mathbf{k}}_r^{(N)}$ and $\mathbf{E}_i^{(N+1)}(\mathbf{r}) = \mathbf{E}_r^{(N)}(\mathbf{r})$. With the analogous process as Eq. (9) and Eq. (10), the (N+1)-th bounce of scattered electric field can be calculated. Sequentially, the scattering induced by the N-th bounce can be calculated by the TDPO integral which is similar to Eq. (5). The difference is the incident electric field should be substituted with $\mathbf{E}_i^{(N)}$.

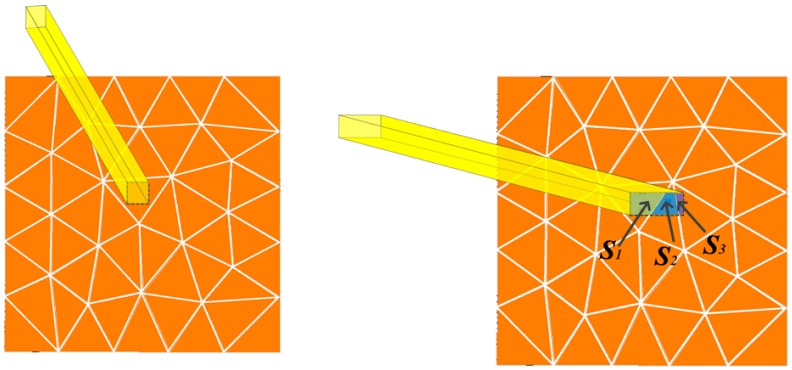
It should be noticed that the wave form in time domain $E_0(t)$ will delay with the multiple bounces. The time delay of the time domain can be deduced by the ray path length. Let $L^{(N)}$ be the ray path length between the N-th and (N-1)-th bounces. The time delay of the N-th bounce can be expressed as $\tau^{(N)} = L^{(N)}/c$.

In our paper, TDSBR is used to obtain the electric field information (including field amplitude, polarization, incident direction and so on), the intersection area information (including the position of the hit points) and the path length when a ray tube hit the target. All the information is needed by TDPO integral equation to obtain the scattering field which is induced by the illumination of this ray tube. This shows the combination of TDSBR and TDPO. The most complex part of the combination is

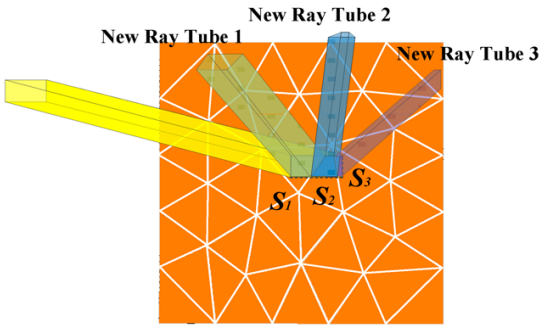
 doi:10.20944/preprints202606.1117.v1

the calculation of the intersect area. Since the ray tube may hit more than one facet at a bounce, the divergence problem should be carefully treated.

It should be noted that when one ray tube encounters the target, there are two inter-section cases: (1) The ray tube intersects with only one facet (Figure 4(a)). In this case, the intersection area is a quadrangle. The intersection points are easy to obtain. With the posi-tions of the four vertexes of the quadrangle, the scattering of this area can be evaluated through TDPO. (2) The ray tube hits the facets' boundaries and intersects with more than one facets (Figure 4(b)). In this case, the ray tube will split according to the shape of the facets of the target and generate new rays with different origins and reflection directions. There are two ways to solve the divergence problem. The first one is discarding the ray tube. This applies to the very small-size ray tube and target whose surface curvature is small. Another way is using the adaptive partition algorithm in ray tracing. The ray tubes emit-ted from ray aperture are relatively large. When the initial beam encounters the target, it splits according to target shape. Each split will generate new ray tubes. The shape of the new ray tubes will change. Each new ray should be retraced. This method will increase the complexity of the problem.



(a) The ray tube interests with one facet (b) The ray tube interests more than one facet



(c) The ray tube split to new rays with new origins and directions

Figure 4. Illustration of the ray tube intersection cases.

Considering the ray tube size is relatively small, e.g. 1/10 of the wavelength, it is unnecessary to consider all the divergence cases. We evaluate the intersection area of the ray tube and each facet. If a ray tube hits n facets, we denote the intersection area of the ray tube and the i-th facet as A_i. Then, the ratio $A_i / \sum_{j=1}^{n} A_j$ is computed. If the ratio is larger than 40%, the new ray would be generated and retraced, and the scattering is calculated. If the ratio is between 20% and 40%, the scattering from this area will be calculated but the ray tube will not be generated. If the ratio is smaller than 20%, the scattering be ignored and the ray tube will not be generated.

We use NVIDIA OptiX to perform ray tracing and electric field calculation within each ray tube. More details about the implementation can be found in our previous work [31].

 doi:10.20944/preprints202606.1117.v1

3.3. Q-Learning for Adaptive Ray Tube Scaling

3.3.1. Problem Formulation

The basic scheme of Q-learning is designing a Markov Decision Process (MDP) which consists of a state space S, an action space A, and a reward function R. The agent observes the state $s \in S$ of the environment and takes an action $a \in A$ according to a policy $\pi(a|s)$, which is a mapping from states to action probabilities. After taking the action, the agent receives a reward r and observes the next state s'. The goal of the agent is to learn a policy that maximizes the expected cumulative reward over time.

In our problem, the agent is responsible for deciding where to generate new ray tubes. The action of the agent is to adjust the ray shooting probability of each grid cell (Figure 5).

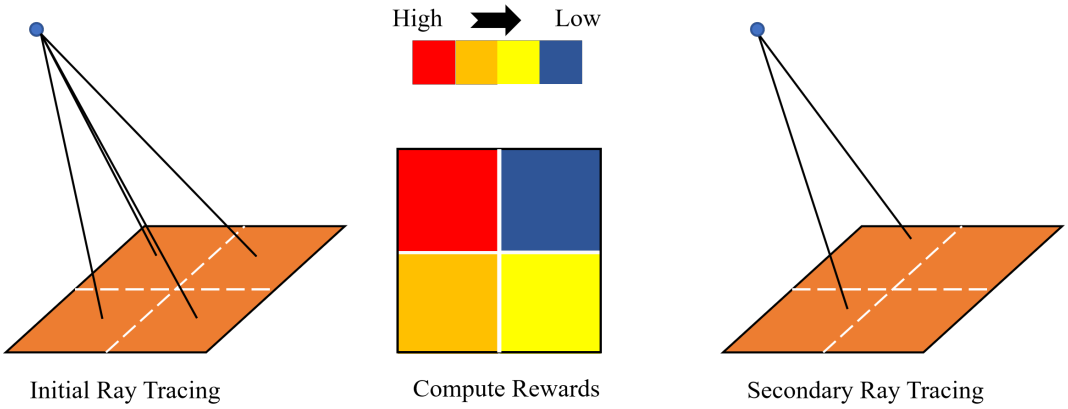


Figure 5. Illustration of the agent's action. We divide the imaging area into grid cells. Each grid cell corresponds to a state in the MDP. In the initial tracing, the agent shoots one ray tube from the radar to the center of each grid cell. Then, the agent observes the state of each grid cell and takes an action to adjust the ray shooting probability for that cell. In the secondary tracing, the agent shoots new ray tubes according to the adjusted shooting probabilities.

To design the reward function, we need to consider what ensures the accuracy of the SBR-based SAR simulation. For SBR-based SAR simulation, there are two main parts to ensure the accuracy: (1) electric field computation; (2) signal processing. For the electric field computation part, the accuracy is mainly affected by the structure of the target. For example, if the target surface is flat, the scattering from this area will be relatively simple. However, if the target surface is complex, e.g., with many small facets or sharp edges, the scattering from this area will be more complex and multiple ray tubes are needed to capture the scattering behavior of this area. If we only consider the electric field computation part, we should design a reward function that encourages the agent to allocate more ray tubes to the complex areas and fewer ray tubes to the flat areas. For the signal processing part, the accuracy is mainly affected by the distances from the radar to the scattering centers. If we only consider the signal processing part, we should design a reward function that encourages the agent to allocate more ray tubes to the areas that can introduce new distances. Hence, we design a combined reward function that considers both the electric field computation and signal processing parts to guide the agent.

As illustrated in Figure 6, each time a ray tube is shot, the distance $d^{(k)}$ and the electric field $E^{(k)}$ are computed. We use standard deviations $std(\cdot)$ of the normalized distances and electric fields as the rewards for the signal processing and electric field computation parts, respectively, where $\cdot$ represents the set of all distances or electric fields computed from the ray tubes shot in the current tracing step.

 doi:10.20944/preprints202606.1117.v1

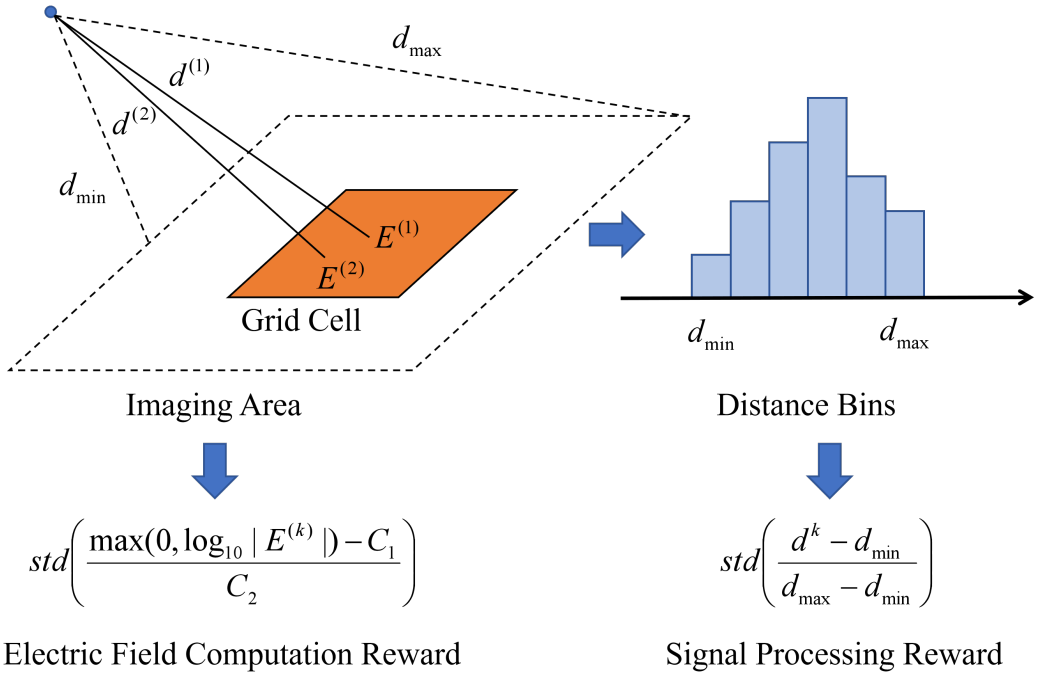


Figure 6. Illustration of the reward computation. Each time a ray tube is shot, the distance $d^{(k)}$ and the electric field $E^{(k)}$ are computed. The standard deviations of the normalized distances and electric fields are used as the rewards for the signal processing and electric field computation parts, respectively.

3.3.2. Constructing Reward Function

The target scene is partitioned into a matrix of discrete spatial bins (grid cells) indexed by (i, j), where the size of each grid cell matches the resolution limits ($\Delta_r \times \Delta_a$). Let us denote the radar position as $\mathbf{P}_r$. The center of the (i, j)-th grid cell is denoted as $\mathbf{C}_{i,j} = (x_i, y_j, 0)$.

For each cell (i, j), the initial primary ray is shot from the radar position $\mathbf{P}_r$ directly toward the cell center $\mathbf{C}_{i,j}$. The ray vector is defined as $\mathbf{k}_0 = (\mathbf{C}_{i,j} - \mathbf{P}_r)/\|\mathbf{C}_{i,j} - \mathbf{P}_r\|$. As this ray tube propagates and impinges upon the target facets contained within or behind this spatial bin, the SBR solver traces its multiple bounces and computes the final scattered electric field at the receiver via Eq. (5). We denote the electric field as:

$$E_{i,j}^{(0)} = \mathcal{F}_{\text{SBR}}(\mathbf{P}_r, \mathbf{C}_{i,j}), \tag{15}$$

where $\mathcal{F}_{\text{SBR}}$ represents the SBR method introduced in Subsection 3.2.

To evaluate whether the geometric features within the cell (i, j) induce complex multi-path scattering or phase cancellation, a stochastic evaluation mechanism is introduced. A set of secondary points $\mathbf{X}_{i,j}^{(k)}$ $(k = 1, 2, \ldots, N)$ are randomly sampled within the boundaries of the grid cell (i, j) using a uniform distribution. Secondary rays are sequentially shot from $\mathbf{P}_r$ toward these random positions, and their corresponding scattered electric fields are denoted as:

$$E_{i,j}^{(k)} = \mathcal{F}_{\text{SBR}}(\mathbf{P}_r, \mathbf{X}_{i,j}^{(k)}).$$

The complex standard deviation of the computed electric fields within the cell is then utilized as an indicator of local scattering complexity. The standard deviation $\sigma_{i,j}$ is calculated as:

$$\bar{E}_{i,j} = \frac{1}{N+1} \sum_{k=0}^{N} E_{i,j}^{(k)} \tag{16}$$

$$std(\{E_{i,j}^{(k)}\}) = \sqrt{\frac{1}{N+1} \sum_{k=0}^{N} \left| E_{i,j}^{(k)} - \bar{E}_{i,j} \right|^2} \tag{17}$$

 doi:10.20944/preprints202606.1117.v1

A high value of $std(\{E_{i,j}^{(k)}\})$ reveals that minor spatial perturbations within the resolution cell lead to drastic fluctuations in the echo amplitude or phase, signifying high geometric complexity. Conversely, a low $std(\{E_{i,j}^{(k)}\})$ indicates a flat or homogenous surface where a single center ray is sufficient to represent the entire cell's scattering behavior.

The magnitudes of electric fields may vary in a very large dynamic range, e.g., from 10^{-6} to 10^{-2}, which makes the smaller electric fields have less effects on the computation of the standard deviation. To visualize a SAR image, we usually set a dynamic range, e.g., 30 dB or 40 dB. That is, the electric fields should be converted to logarithmic scale and then normalized by the dynamic range. However, the maximum and mimimum magnitudes of the electric fields are not known before the simulation. They depend on the target geometry and material properties, the radar position and so on. To solve this problem, we can use the electric fields computed from the primary rays as the reference for normalization. The normalized electric fields can be expressed as:

$$\hat{E}_{i,j}^{(k)} = \frac{\log_{10}|E_{i,j}^{(k)}| - C_1}{C_2}, \tag{18}$$

where $C_1 = \log_{10}\max_{i,j}|E_{i,j}^{(0)}|$ and $C_2 = 10^{\text{dynamic range in dB}/10}$. Then, the standard deviation can be computed based on the normalized electric fields $\hat{E}_{i,j}^{(k)}$.

The distance bins are constructed by three parameters, i.e., the minimum range $r_{\min}$, the maximum range $r_{\max}$ and the range resolution Δ_r. $r_{\min}$ and $r_{\max}$ can be calculated based on the radar flight trajectory and the imaging area. It is a constrained convex optimization problem. Without loss of generality, suppose a rectangular imaging area centered at the origin of the XOY plane. In our implementation, we segment flight trajectory according to the pulse repetition frequency (PRF) and compute the minimum and maximum distances from the radar at each position to the four edges of the rectangular area. To compute the minimum and maximum distances from a point to a line segment, we only need to compute and compare three distances: (1) two distances from the point to the two endpoints of the line segment; (2) distance from the point to the infinite line defined by the line segment. The process can be parallelized for all radar positions.

Δ_r is the sampling interval in the fast-time domain, which is determined by the bandwidth B of the LFM signal. According to Nyquist sampling theorem, Δ_r should be less than $(r_{\max} - r_{\min})/2B$.

For different scenarios, the ranges of distances are different. To make the reward function more general, we can normalize the distances by the maximum range $r_{\max}$, i.e., $\hat{d}^{(k)} = d^{(k)}/r_{\max}$. Then, the standard deviation of the normalized distances can be computed similarly as Eq. (17).

The final reward function is given by:

$$R_{i,j} = -\alpha \cdot \hat{n}_{i,j}^{(k)} + \beta_r \cdot std(\hat{r}_{i,j}^{(k)}) + \beta_E \cdot std(\hat{E}_{i,j}^{(k)}), \tag{19}$$

where α, β_r, β_E are three positive constants. $\hat{n}_{i,j}^{(k)}$ is the normalized number of rays shot in the cell (i,j), which can be expressed as $\hat{n}_{i,j}^{(k)} = n_{i,j}^{(k)}/n_{\max}$, where $n_{i,j}^{(k)}$ is the number of rays shot in the cell (i,j) and $n_{\max}$ is the maximum number of rays that can be calculated by the grid cell size divided by the ray tube size. The first term in Eq. (19) is the penalty for computation costs since more rays means more computations in ray tracing and signal processing. The second term is the reward for improving the signal processing accuracy since a higher standard deviation of distances indicates more new distance information is introduced by the new rays. The third term is the reward for improving the electric field computation accuracy since a higher standard deviation of electric fields indicates more complex scattering behavior is captured by the new rays.

3.3.3. Q-Learning-Based Ray Generation

The Q-learning-based ray generation method is shown in Algorithm 1.

 doi:10.20944/preprints202606.1117.v1

Algorithm 1 Q-learning-based ray generation method

```
Initialize Q-table Q(s, a) with zeros for all state-action pairs.
Initialize ray shooting probability P_{i,j} = 0.9 for all grid cells.
Initial ray tracing: Shoot one ray tube from the radar to the center of each grid cell.
while not converged do
    for each grid cell (i, j) do
        Observe the current state s_{i,j}
        Select an action a_{i,j}.
        Shoot new rays according to the updated probability P_{i,j}.
        Compute the reward R_{i,j} based on the new electric fields and distances.
        Update the Q-table.
    end for
end while
```

1) State Space (S)

The state $s_{i,j}$ of the cell (i,j) is determined by quantizing its current scattering standard deviation $std(\{\hat{E}_{i,j}^{(k)}\})$ and $std(\{\hat{r}_{i,j}^{(k)}\})$ into discrete levels. As electric fields and distances are normalized, the standard deviations are in $[0,1]$. We can discretize this range into M equal intervals, resulting in a $M \times M$ state space for each cell.

2) Action Space (A)

The action $a_{i,j}$ corresponds to adjusting the ray sampling probability $P_{i,j}$ of the cell for the next aperture simulation step:

- a_0 (Decrease): Reduce the sampling probability ($P_{i,j} \leftarrow \max(P_{\min}, P_{i,j} - \Delta P)$).
- a_1 (Maintain): Keep the current sampling probability unchanged.
- a_2 (Increase): Raise the sampling probability ($P_{i,j} \leftarrow \min(P_{\max}, P_{i,j} + \Delta P)$).

We set ΔP as a fixed increment, e.g., 0.05. $P_{\min}$ and $P_{\max}$ are the lower and upper bounds of the sampling probability, e.g., 0.0 and 1.0, respectively.

3) Reward Function (R)

The reward function is given by Eq. (19).

4) Q-Table Update Rule

The Q-values, representing the expected long-term rewards of taking specific actions under certain geometric states, are updated iteratively via the temporal difference (TD) learning rule:

$$Q(s_{i,j}, a_{i,j}) \leftarrow Q(s_{i,j}, a_{i,j}) + \eta \left[R_{i,j} + \gamma \max_{a'} Q(s'_{i,j}, a') - Q(s_{i,j}, a_{i,j}) \right] \tag{20}$$

where η is the learning rate, and γ is the discount factor.

During the continuous synthetic aperture simulation, as the platform moves to Y_{m+1}, the framework checks the updated probability map $P_{i,j}$. Cells with lower probabilities will bypass the secondary random ray-tracing step, directly adopting the previous aggregated electric fields as the cell's total contribution, thereby drastically compressing the total number of simulated rays across the full aperture imaging chain.

3.3.4. Termination Condition

The Q-learning process iteratively continues until the change in the Q-table values falls below a predefined threshold ϵ across all state-action pairs, indicating convergence to an optimal policy for ray tube allocation.

3.4. Implementation Details

We would like to emphasize three important implementation details of the proposed method.

1) Phase issues

 doi:10.20944/preprints202606.1117.v1

To avoid redundant phase accumulation, the phase variations induced by spatial propagation delay are excluded from the high-frequency electric field calculation in the SBR phase. Since these propagation-related phase shifts are already inherently included in the LFM signal model in Eq. (1), the SBR module focuses solely on capturing amplitude attenuation and the scattering/polarization phase jumps determined by the target geometry and material properties. On the other hand, one can remove the propagation phase shifts from the LFM signal model, i.e. $\exp(j2\pi f_c(t-\tau_i))$ in Eq. (1), and keep the SBR module unchanged.

2) Precision issues

Maintaining the precision of these propagation-induced phase variations is important for SAR imaging, but it poses a challenge for standard GPU hardware. The NVIDIA OptiX ray-tracing engine utilizes single-precision floating-point numbers for intersection testing. When simulating scenarios with large operational ranges, the limited precision introduces significant spatial quantization errors. These small coordinate deviations lead to severe, chaotic phase errors during LFM echo synthesis.

To mitigate this issue, we implemented a hybrid double-precision intersection refinement module. Once NVIDIA OptiX returns the primitive intersection data (i.e., the hit distance and the corresponding triangle facet index), the exact intersection coordinates are re-evaluated in double precision using the ray vector and the vertices of the hit triangle. Although performing these double-precision floating-point operations introduces a minor computational overhead, it is indispensable for eliminating phase artifacts and ensuring the fidelity of the simulated SAR images.

3) Parallelization

By utilizing the RT cores of NVIDIA RTX GPUs, the ray tracing is much faster than the subsequent signal processing. Compared with performing 25×25 ray tracing for 16 times, performing 100×100 ray tracing for one launch has incremental overall performance improvement. As our Q-learning-based ray generation method needs to iteratively determine whether to generate new rays within each grid cell, more iterations generally lead to more ray number reduction. Therefore, there's a trade-off between the number of iterations and the number of rays per launch. We also need to consider the hardware limitations, such as GPU memory and the maximum number of rays that can be traced in parallel. For instance, if the targeting imaging area is $100m \times 100m$ and the carrier frequency is 3 GHz, the ray tube size will be 0.01 m, which means the total number of rays will be 10^8. As we perform double-precision electric field calculations within each ray tube, a large number of rays may lead to out-of-memory issues.

4) Iterative standard deviation computation trick

As there may be a large number of rays within each grid cell or distance bin in every launch, computing the standard deviation of the electric fields or distances can be computationally expensive. Hence, we use a constant standard deviation for all rays in each launch. For example, if two rays are in the same distance bin, they will share the same standard deviation of distances in the previous launch. The standard deviation of distances is updated iteratively by Eq. (21):

$$\begin{cases} \mu_{new} = \dfrac{n_{old}}{n_{new}+n_{old}}\mu_{old} + \dfrac{1}{n_{new}+n_{old}}\sum d_{new} \\ \sigma_{new} = \sqrt{\dfrac{n_{old}}{n_{new}+n_{old}}\sigma_{old}^2 + \dfrac{1}{n_{new}+n_{old}}\sum (d_{new}-\mu_{new})^2}, \end{cases} \tag{21}$$

where σ_{old} and μ_{old} are the standard deviation and mean of the distances of the previous n_{old} rays, respectively, and d_{new} is the distance of the new ray tube. That is, we do not need to access all the distances of the previous ray tubes to compute the standard deviation. The updating formula of the standard deviation of electric fields is similar to Eq. (21).

4. Experiments

All our experiments are performed on a workstation equipped with an NVIDIA RTX 4090 GPU and an Intel i9-14900K. The GPU driver version is 595.79, CUDA version is 12.9 and OptiX version is 9.1.

 doi:10.20944/preprints202606.1117.v1

4.1. Parameter Settings

In the LFM signal model, the carrier frequency f_c is set to 12 GHz, the bandwidth B is set to 150 MHz, the antenna length L is set to 1 m, the pulse duration T is set to 1 μs, the flight speed v is set to 100 m/s and the pulse repetition frequency (PRF) is set to 1 kHz.

In SBR module, we use the carrier frequency f_c in SBR module to calculate the electric field for simplicity. The ray tube size is set to 1/10 of the wavelength λ at the carrier frequency. The tracing depth is set to 3, which means only 3 bounces are considered.

In the Q-learning module, the state discrete level M is set to 10, the learning rate η is set to 0.01, the discount factor γ is set to 0.9. The penalty coefficient α is set to 1 and the accuracy weights β_r and β_E are set to 10.

4.2. Evaluation Metrics

The proposed method aims to accelerate SBR-based SAR imaging simulation for computer vision tasks, such as data augmentation and adversarial attacks. Therefore, we use the structural similarity index measure (SSIM) [32] to evaluate the similarity between SAR images generated by the original SBR method (SBR_o) and the proposed method (SBR_p). Higher SSIM values indicate greater image similarity, and the maximum value of 1 indicates two identical images.

As data augmentation and adversarial attack methods are usually used in object detection tasks, we also use the mean average precision (mAP) of a fine-tuned state-of-the-art object detection model, i.e., YOLOv12 [33], to evaluate detection performance changes. YOLOv12 is built on PyTorch 2.2.2 which relies on CUDA 12.1. Hence, there are some compatibility issues when using YOLOv12 with our CUDA 12.9 environment. To solve this issue, we use Docker to create a compatible environment for YOLOv12. We fine-tune the YOLOv12 model on 3000 images generated by SBR_o and test the model on 1000 images generated by SBR_o and SBR_p, respectively. The LFM signal parameters are kept identical, and only the radar flight path is varied when generating SAR images. A small mAP difference indicates that the proposed method can generate SAR images with performance comparable to the original method for YOLOv12.

In addition to accuracy metrics, we compare the average computation-time reduction ratio and average ray-count reduction ratio between SBR_o and SBR_p to evaluate efficiency gains. These reduction ratios are computed for each pair of SAR images generated by SBR_o and SBR_p under the same target, radar flight path, and parameter settings. Because computation time and ray count depend strongly on target geometry, flight path, and LFM parameters, their absolute values can vary substantially. Therefore, reduction ratios provide a more stable and comparable measure of efficiency improvement across different scenarios.

4.3. Results and Analysis

Some simulation results are shown in Figure 7. The images generated by SBR_o exhibit higher contrast because they involve more rays and therefore yield more accurate echo calculations.

Across 1000 pairs of SAR images generated by SBR_o and SBR_p, the average SSIM is 0.983, indicating that the proposed method preserves structural image similarity. The mAP of YOLOv12 on SAR images generated by SBR_o and SBR_p is 0.951 and 0.950, respectively, showing that SBR_p performs comparably to SBR_o for object-detection-oriented data augmentation and adversarial attack tasks. The average ray-count reduction ratio is 56.9%, indicating that the proposed method reduces the number of rays by more than half. The average computation time reduction ratio is 61.2%. The computation-time reduction ratio is higher than the ray-count reduction ratio, indicating that computational cost is not linearly proportional to the number of rays. This is mainly because GPU cores and memory bandwidth are limited, so a massive number of rays increases overall computation time in a nonlinear manner.

 doi:10.20944/preprints202606.1117.v1

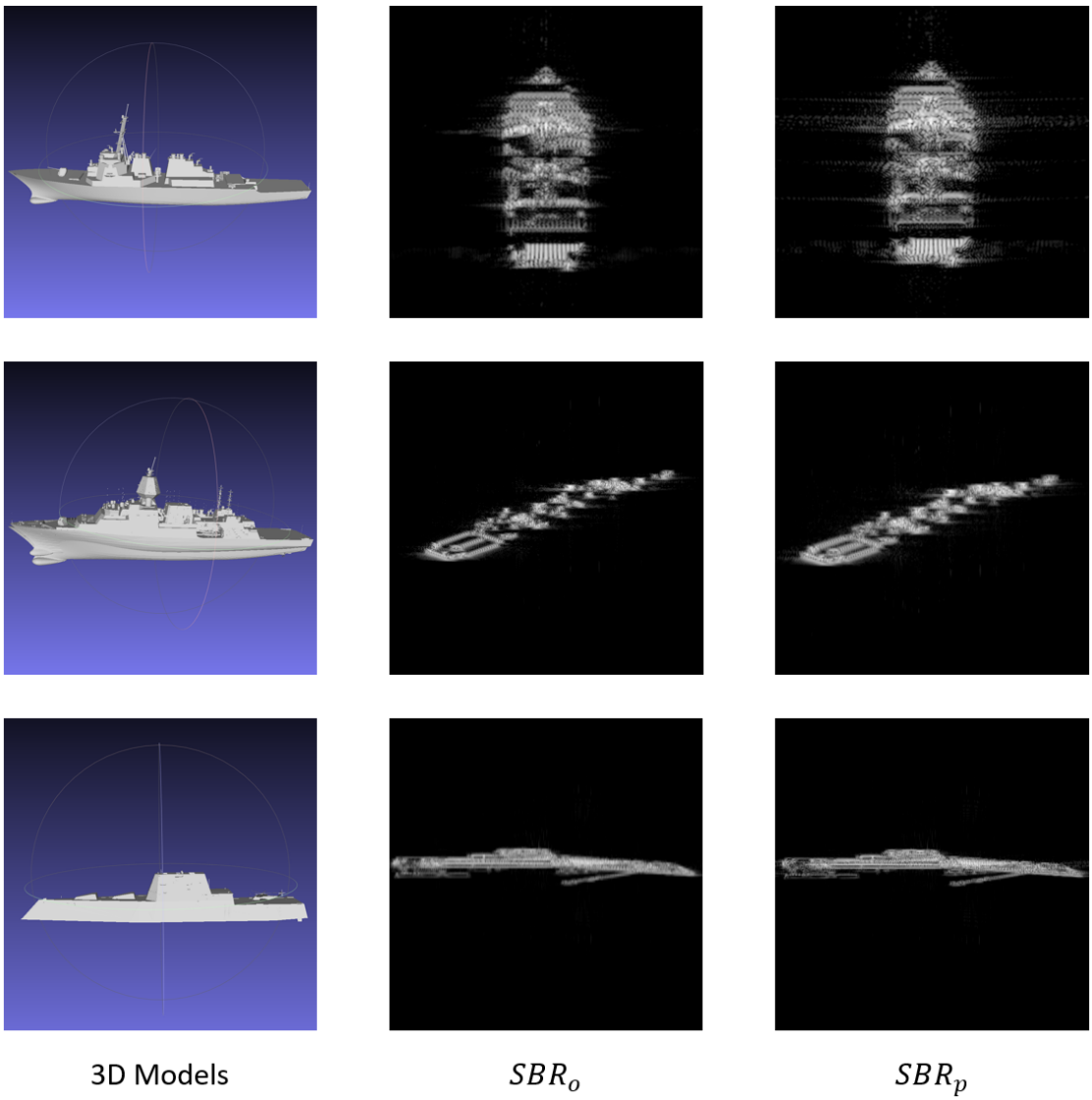
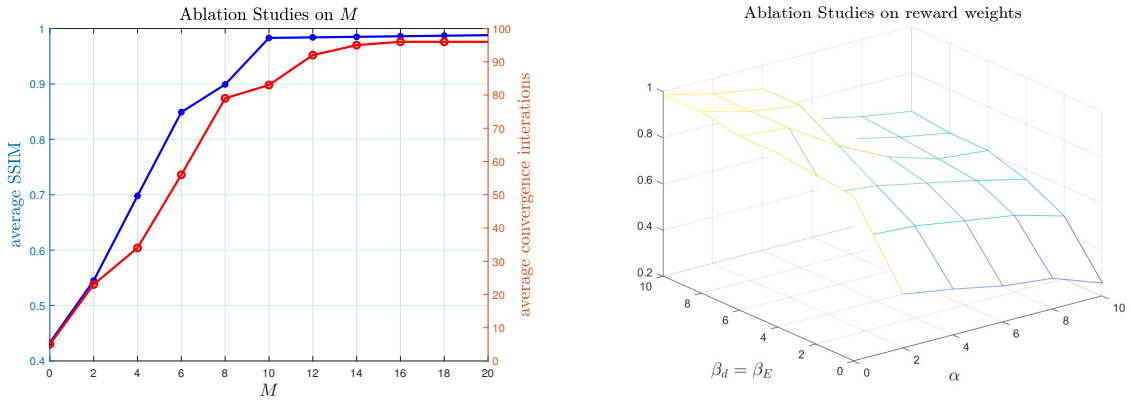


Figure 7. SAR images generated by the original SBR method (SBR_o) and the proposed SBR method (SBR_p).

4.4. Ablation Studies

Some hyperparameters in Q-learning have significant impacts on the performance of the proposed method. Here, we conduct ablation studies on two groups of hyperparameters: (1) the state discrete level M; (2) the reward weights α, β_r and β_E. We use SSIM as the metric to evaluate the image similarity between SBR_o and SBR_p under different hyperparameter settings. The results are shown in Figure 8. As shown in Figure 8(a), the image similarity increases with the state discrete level M because a higher M allows the agent to learn a more fine-grained policy for ray tube allocation. However, a higher M also increases the state space and therefore increases the training time of the Q-learning algorithm. As shown in Figure 8(b), the image similarity is higher when the reward weights are higher, e.g., $\alpha = 1$, $\beta_r = 10$ and $\beta_E = 10$. When α is too high, the agent is more likely to reduce the ray shooting probability, which may lead to insufficient rays and therefore lower image similarity. When β_r or β_E is too high, the agent is more likely to increase the ray shooting probability, which may lead to excessive rays and therefore higher computation time. Therefore, it is important to balance the reward weights to achieve a good trade-off between image similarity and computation time reduction.

 doi:10.20944/preprints202606.1117.v1



(a) (b)

Figure 8. Ablation studies on (a) the state discrete level M and (b) the reward weights α, β_r and β_E.

5. Conclusion

In this letter, we proposed a method to accelerate SBR-based SAR imaging simulation using Q-learning. The experimental results demonstrate that the proposed method can generate SAR images with high similarity to those produced by the original SBR method while significantly reducing the computation time. This makes the method suitable for computer vision tasks such as data augmentation and adversarial attacks in SAR image analysis. The proposed method is a framework that can be extended to other ray-tracing-based electromagnetic simulation methods, SAR imaging methods (e.g. chirp scaling) and reinforcement learning algorithms.

Author Contributions: Conceptualization, Dayong Tian; methodology, Dayong Tian; software, Dayong Tian and Shuo Wang; validation, Shuo Wang and Md. Gazi Salahuddin; writing—original draft preparation, Dayong Tian and Md. Gazi Salahuddin; writing—review and editing, Md. Gazi Salahuddin and Xiaoyang Li; project administration, Xiaoyang Li; funding acquisition, Xiaoyang Li.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: During the preparation of this manuscript/study, the author(s) used [DeepSeek, V3.2] for the purposes of converting hand-writing formulas to LaTeX, formatting tables, equations and figures in LaTeX, and English grammar checking. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Cui, J.; Duan, J.; Guo, W.; Peng, C.; Li, H. SAR-ESAE: Echo Signal-Guided Adversarial Example Generation Method for Synthetic Aperture Radar Target Detection. *Remote Sensing* **2025**, *17*. https://doi.org/10.3390/rs17173080.
2. Tong, Y.; Xiong, K.; Liu, J.; Cao, G.; Fan, X. MAIENet: Multi-Modality Adaptive Interaction Enhancement Network for SAR Object Detection. *Remote Sensing* **2025**, *17*. https://doi.org/10.3390/rs17233866.
3. Li, L.; Huang, L.; Meng, T.; Xing, C.; Yang, T.; Li, W.; Lu, P. MFE-STN: A Versatile Front-End Module for SAR Deception Jamming False Target Recognition. *Remote Sensing* **2025**, *17*. https://doi.org/10.3390/rs17233848.
4. Cui, Y.; Liu, Z.; Ruan, L.; Sheng, B.; Wang, N.; Xiao, X.; Bian, X. An Azimuth-Continuously Controllable SAR Image Generation Algorithm Based on GAN. *Remote Sensing* **2025**, *17*. https://doi.org/10.3390/rs17223763.
5. Li, J.; Meng, W.; Chai, S.; Guo, L.; Xi, Y.; Wen, S.; Li, K. An Accelerated Hybrid Method for Electromagnetic Scattering of a Composite Target–Ground Model and Its Spotlight SAR Image. *Remote Sensing* **2022**, *14*. https://doi.org/10.3390/rs14246332.

 doi:10.20944/preprints202606.1117.v1

6. Suk, S.; Seo, T.I.; Park, H.S.; Kim, H.T. Multiresolution grid algorithm in the SBR and its application to the RCS calculation. *Microwave and Optical Technology Letters* **2001**, *29*, 394–397, [https://onlinelibrary.wiley.com/doi/pdf/10.1002/mop.1188]. https://doi.org/https://doi.org/10.1002/mop.1188.
7. Jin, K.S.; Suh, T.I.; Suk, S.H.; Kim, B.C.; Kim, H.T. Fast Ray Tracing Using A Space-Division Algorithm for RCS Prediction. *Journal of Electromagnetic Waves and Applications* **2006**, *20*, 119–126. https://doi.org/10.1163/156939306775777341.
8. Bang, J.K.; Kim, B.C.; Suk, S.H.; Jin, K.S.; Kim, H.T. Time Consumption Reduction of Ray Tracing for Rcs Prediction using Efficient Grid Division and Space Division Algorithms. *Journal of Electromagnetic Waves and Applications* **2007**, *21*, 829–840. https://doi.org/10.1163/156939307780749129.
9. Huo, J.; Xu, L.; Shi, X.; Yang, Z. An Accelerated Shooting and Bouncing Ray Method Based on GPU and Virtual Ray Tube for Fast RCS Prediction. *IEEE Antennas and Wireless Propagation Letters* **2021**, *20*, 1839–1843. https://doi.org/10.1109/LAWP.2021.3098970.
10. Zhao, L.; Li, J.; Meng, W.; Guo, L.X.; Xi, Y.J. An Improved SBR Method Based on N-URD Emission Plane and Forward Bounding Box for Fast RCS Prediction. *IEEE Antennas and Wireless Propagation Letters* **2026**, *25*, 746–750. https://doi.org/10.1109/LAWP.2025.3638742.
11. Hu, S.; Guo, L.X.; Liu, Z.; Zhong, Z.; Nan, Z. A High-Performance GPU-Accelerated Ray-Tracing Method for Real-Time V2V Channel Modeling. *IEEE Antennas and Wireless Propagation Letters* **2025**, *24*, 2527–2531. https://doi.org/10.1109/LAWP.2025.3567499.
12. Kajiya, J.T. The rendering equation. *SIGGRAPH Comput. Graph.* **1986**, *20*, 143–150. https://doi.org/10.1145/15886.15902.
13. Yan, R.; Guo, H.; Huang, L.; Xiao, N.; Li, S.; Wang, Y.; Lv, Y.; Chen, G. A Survey on Deep Learning for Monte Carlo Path Tracing. *ACM Comput. Surv.* **2025**, *58*. https://doi.org/10.1145/3768618.
14. Dahm, K.; Keller, A. Learning Light Transport the Reinforced Way. In Proceedings of the ACM SIGGRAPH 2017 Talks, New York, NY, USA, 2017. https://doi.org/10.1145/3084363.3085032.
15. Zheng, Q.; Zwicker, M. Learning to Importance Sample in Primary Sample Space. *Computer Graphics Forum* **2018**, *38*.
16. Bako, S.; Meyer, M.; DeRose, T.; Sen, P. Offline Deep Importance Sampling for Monte Carlo Path Tracing. *Computer Graphics Forum* **2019**, *38*, 527–542. https://doi.org/10.1111/cgf.13858.
17. Müller, T.; McWilliams, B.; Rousselle, F.; Gross, M.; Novák, J. Neural Importance Sampling. *ACM Transactions on Graphics* **2019**, *38*. https://doi.org/10.1145/3341156.
18. Müller, T.; Rousselle, F.; Keller, A.; Novák, J. Neural Control Variates. *ACM Transactions on Graphics* **2020**, *39*. https://doi.org/10.1145/3414685.3417804.
19. Zhu, S.; Xu, Z.; Sun, T.; Kuznetsov, A.; Meyer, M.; Jensen, H.W.; Su, H.; Ramamoorthi, R. Photon-Driven Neural Reconstruction for Path Guiding. *ACM Transactions on Graphics* **2021**, *41*, 1–15.
20. Zhu, S.; Xu, Z.; Sun, T.; Kuznetsov, A.; Meyer, M.; Jensen, H.W.; Su, H.; Ramamoorthi, R. Hierarchical Neural Reconstruction for Path Guiding Using Hybrid Path and Photon Samples. *ACM Transactions on Graphics* **2021**, *40*, 1–16.
21. Dong, H.; Wang, G.; Li, S. Neural Parametric Mixtures for Path Guiding. In Proceedings of the ACM SIGGRAPH 2023 Conference Proceedings, New York, NY, USA, 2023.
22. Huang, J.; Iizuka, A.; Tanaka, H.; Komura, T.; Kitamura, Y. Online Neural Path Guiding with Normalized Anisotropic Spherical Gaussians. *ACM Transactions on Graphics* **2024**, *43*. https://doi.org/10.1145/3649310.
23. Lu, H.; Chang, W.; Hedstrom, T.; Li, T.M. Real-Time Path Guiding Using Bounding Voxel Sampling. *ACM Transactions on Graphics* **2024**, *43*. https://doi.org/10.1145/3658203.
24. Dachsbacher, C. Analyzing Visibility Configurations. *IEEE Transactions on Visualization and Computer Graphics* **2011**, *17*, 475–486. https://doi.org/10.1109/TVCG.2010.77.
25. Kuznetsov, A.; Kalantari, N.K.; Ramamoorthi, R. Deep Adaptive Sampling for Low Sample Count Rendering. *Computer Graphics Forum* **2018**, *37*.
26. Vogels, T.; Rousselle, F.; McWilliams, B.; Röthlin, G.; Harvill, A.; Adler, D.; Meyer, M.; Novák, J. Denoising with Kernel Prediction and Asymmetric Loss Functions. *ACM Transactions on Graphics* **2018**, *37*, 1–15.
27. Huo, Y.; Wang, R.; Zheng, R.; Xu, H.; Bao, H.; Yoon, S.e. Adaptive Incident Radiance Field Sampling and Reconstruction Using Deep Reinforcement Learning. *ACM Transactions on Graphics* **2020**, *39*, 1–17.
28. Hasselgren, J.; Munkberg, J.; Salvi, M.; Patney, A.; Lefohn, A.E. Neural Temporal Adaptive Sampling and Denoising. *Computer Graphics Forum* **2020**, *39*.

 doi:10.20944/preprints202606.1117.v1

29. Salehi, F.; Manzi, M.; Roethlin, G.; Weber, R.; Schroers, C.; Papas, M. Deep Adaptive Sampling and Reconstruction Using Analytic Distributions. *ACM Transactions on Graphics* **2022**, *41*. https://doi.org/10.1145/3550454.3555515.
30. Firmino, A.; Frisvad, J.R.; Jensen, H.W. Denoising-Aware Adaptive Sampling for Monte Carlo Ray Tracing. In Proceedings of the ACM SIGGRAPH 2023 Conference Proceedings, New York, NY, USA, 2023. https://doi.org/10.1145/3588432.3591537.
31. Wei, Y.; Tian, D.; Li, J.; Wang, J.; Chai, S.; Guo, L. Efficient GPU implementation of the time-domain shooting and bouncing rays method on electrically large complex target. *Waves in Random and Complex Media* **2025**, *35*, 4741–4760. https://doi.org/10.1080/17455030.2022.2064559.
32. Wang, Z.; Bovik, A.; Sheikh, H.; Simoncelli, E. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* **2004**, *13*, 600–612. https://doi.org/10.1109/TIP.2003.819861.
33. Tian, Y.; Ye, Q.; Doermann, D. YOLOv12: Attention-Centric Real-Time Object Detectors. *arXiv preprint arXiv:2502.12524* **2025**.