Detailed Simulation of Signal-Level Sensor Data Using Monte Carlo Path Tracing and Photon Mapping

DETAILED SIMULATION OF SIGNAL-LEVEL SENSOR DATA USING MONTE CARLO PATH TRACING AND PHOTON MAPPING

BY

DAVID SCHONBORN, B.Eng.

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

Master of Applied Science (2018)	McMaster University
(Electrical & Computer Engineering)	Hamilton, Ontario, Canada

TITLE:	Detailed Simulation of Signal-Level Sensor Data Using Monte Carlo Path Tracing and Photon Mapping
AUTHOR:	David Schonborn B.Eng., (Software Engineering) McMaster University, Hamilton, Canada
SUPERVISOR:	Dr. T. Kirubarajan
NUMBER OF PAGES:	xiii, 54

To my family Julia, Al, and Joanna Schonborn who instilled in me a love of problem solving and discovery at a young age, and to my late friend Moody Younes who inspired me to chase my dreams.

Abstract

Simulated sensor data from active and passive sensors has numerous applications in target detection and tracking. Simulated data is particularly useful in performance evaluation of target tracking algorithms where the ground truth of a scenario must be known. For real sensor data it is impossible to know the ground truth so simulated data must be used.

This paper discusses existing methods for simulation of data from active sensors and proposes a method that builds on existing techniques from the field of computer graphics. An extension to existing methods is proposed to accommodate the simulation of active sensor data for which timing and frequency information is required in addition to intensity. Results from an existing method of active sensor data simulation are compared to the results of the proposed method. Additionally, a cloud computing framework is proposed and its scalability evaluated to address the fairly large computational load of such a simulation.

Acknowledgements

I am fortunate to have a wealth of family and friends, thank you all for your endless support.

I am grateful to have a supervisor who is a legend in the field but is still down to earth. Thank you Dr. Kiruba for introducing me to research in estimation, tracking, and information fusion, for providing an environment where I can thrive, and for always being understanding.

All of the students in our group have access to an indispensable postdoc who can provide insight into practically any problem. Although I'm sure he is very busy, he is never too busy to help me out. Thank you Dr. Thamas for all your help in advancing my understanding and knowledge.

Thank you Dr. Mike McDonald for providing direction and input on my research to ensure that the result would be something practical and useful.

Finally, thank you Cheryl Gies for picking up the slack for all of us who aren't nearly as organized as you are, I don't know how our department would operate without you.

Contents

List of Figures

Chapter 1

Introduction

1.1 Sensor Data in Target Tracking Applications

Sensors such as cameras and radars are commonly used in real world target tracking applications. Sensors capture raw data at the signal level, which is then used as input data for a detection algorithm. The raw data is passed through a detection algorithm which yields detections (or measurements). The detections are used as input data for a tracking algorithm such as a Kalman Filter, Extended Kalman Filter, Particle Filter, etc. The tracking algorithm (tracker) yields estimates of the states of detected targets, which is the goal of target tracking. Figure 1.1 shows how a simple target tracking framework using real sensor data uses the signal captured from the sensors to track targets.

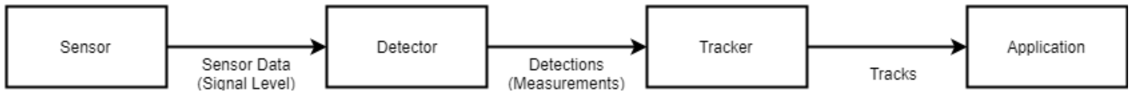

Figure 1.1: A simple tracking framework using real sensor data. Note that the ground truth is not known when capturing real data.

1.2 Simulated Sensor Data in Target Tracking Applications

There are several reasons to simulate sensor data for target tracking applications, mostly related to testing and performance evaluation of target tracking or detection methods. Active research areas in target tracking and detection methods include leveraging prior information about a particular scenario to improve tracking performance (for example (1)), but it may be difficult to obtain real sensor data that is consistent with these particular scenarios, especially for multiple sensor types. Additionally, for the purposes of performance evaluation it is often required that the ground truth of a scenario is known (2). It is not generally possible to know the ground truth for a data set captured from real sensors. It is possible in some cases to manually label the ground truth of a data set, but this process is laborious and error-prone. By simulating sensor data instead of capturing data using real sensors it is possible to address these issues. Figures 1.2 and 1.3 show how simulated data (at the signal level and detection level, respectively) is used in a tracking framework to evaluate performance of tracking methods. Simulated data at the signal level can also be used to evaluate detector performance.

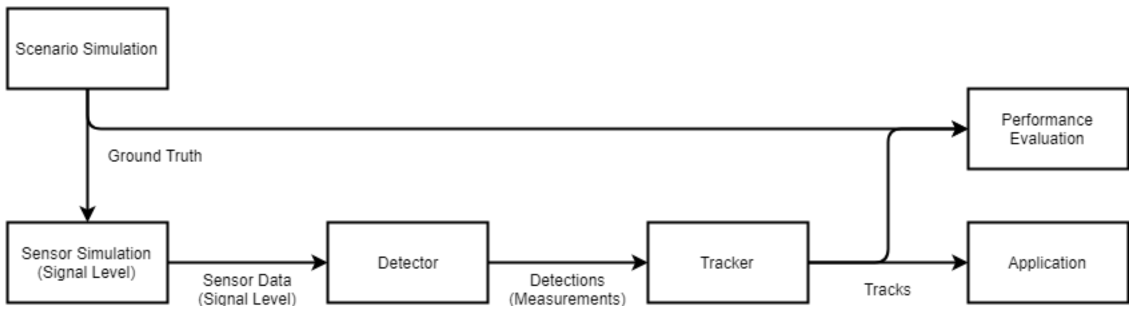

Figure 1.2: A simple tracking framework using sensor data simulated at the signal level to evaluate performance of a tracking or detection method.

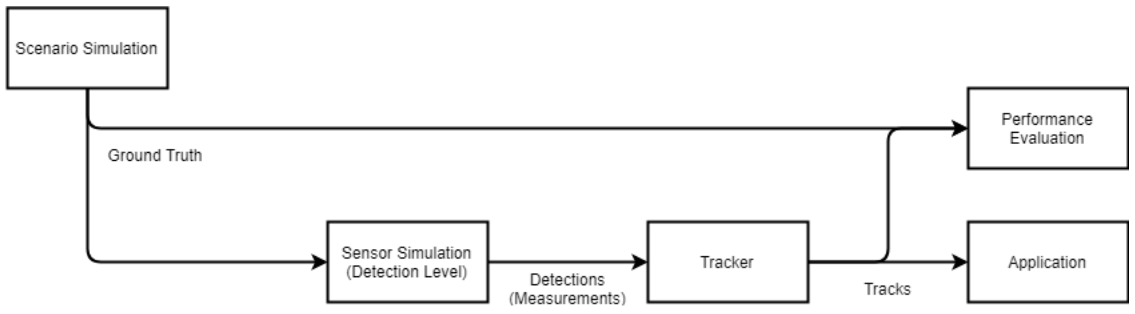

Figure 1.3: A simple tracking framework using sensor data simulated at the detection level to evaluate performance of a tracking method.

1.3 State of the Art and Limitations

Similar simulation methods exist in the literature for various use cases. This section reviews several methods and identifies their strengths and limitations.

1.3.1 Radar Simulation Using The Blender Renderer

In (3) radar data is simulated using Blender. Blender is an open source software suite which includes a ray tracing 3D renderer (4). In this method the output from the Blender renderer is processed to produce a range-velocity-spectrum for objects in the rendered scene (3). The Blender renderer output includes depth and velocity information (using the Z-pass and Vector pass (5)), but is only modelled correctly for the object nearest to the receiver along a given line (3). The authors therefor identify two limitations with their method. A real radar signal may penetrate an object, yielding a return pulse from objects which are occluded by the nearest object, but using this method only the nearest object would generate a return pulse (3). Similarly, in the case of multipath propagation these parameters would only be considered for the final leg of the path taken from the transmitter to the receiver (3). Due to these limitations a range-velocity-spectrum can be computed but will in some scenarios be inaccurate. Additionally, because phase information is not included, the simulated return signal will not account for interference. See section 3.2.5 for a direct comparison between this method and the proposed method.

1.3.2 LIDAR Simulation Using Photon Mapping

A method for simulating LIDAR data using Photon Mapping is presented in (6). Similar to the proposed method, this method introduces another parameter to track the total travel time for the illumination from the transmitter to the receiver (6). Using this timing information the signal returns are quantized into bins based on the sampling frequency provided by the user, yielding a time-gated photon count (which can be used to estimate the returned signal over time) (6). This methodology addresses one of the limitations present in the method described in section 1.3.1 by correctly modelling timing/range information in a scenario with multipath propagation, including signal transmission through objects. Frequency and phase information are still not tracked however, so again it is not possible to account for signal interference and Doppler shifts in the return signal.

1.3.3 Ray-Tracing Simulator for Radar Signals

A ray-tracing method for simulating radar signals is presented in (7). This method accounts for multipath propagation including timing/range information as well as Doppler shift, but does not account for phase (which is listed as a future development goal) (7). Additionally, this method assumes targets to be point targets, each with an assigned value for radar cross section (7). In reality the radar cross section of a target is complex and depends on many factors including aspect angle (8) so this method will not accurately model these effects. Additionally, the points target assumption by definition precludes the possibility of simulations including extended targets.

1.3.4 RaySAR 3D SAR Simulator

RaySAR is a SAR simulator that is reasonably full-featured and is an extended version of the POV-Ray ray tracer (9). This simulator includes limited support for multipath propagation (global illumination is listed as an upcoming extension) (9). Unlike the methods described in sections 1.3.1 through 1.3.3 this simulator does not generate a radar signal or a range-Doppler spectrum, but directly generates a SAR image (9). While this is desirable for SAR simulation, it is not capable of simulating sensor data for other use cases.

1.3.5 Ray-Tracing Simulator for Digital Broadcast Systems

A ray-tracing method for simulating broadcast signals is presented in (10) that is also applicable in the domain of target tracking. This method is similar to the proposed method presented in this paper (see section 3) in that it computes phase information, but different in that it does not track surface velocity during the ray-tracing stage (10). Without this information it is not possible to compute the Doppler shift of signal returns from each propagation path, which is important for applications in target tracking.

1.3.6 A Signal Level Simulator for Multistatic and Netted Radar Systems

In (11) a method for signal level simulation that does not use ray tracing is presented. Similarly to the proposed method, this method simulates data at the signal level and accounts for phase shifts and Doppler shifts information in the signal returns from an arbitrary number of scatters (11). Unlike the proposed method, this method models all reflective objects in the scene as point scatters (except for one surface) (11). This surface that can be included in the model reflects energy and produces additional returns from scatters (for example, specular reflection off the ground plane) (11). This way of modeling objects and the ground plane can be a limitation since in reality there may be multipath propagation from various objects in the scene other than a single surface, and objects in the scene may not be possible to accurately model as point scatters.

1.3.7 Simulation of Radar Signal Propagation via Multipath

In (12) a computationally efficient method for simulating radar signal multipath propagation is described. This method considers ”dominant multipath” only in an effort to minimize computation time, which is described as a signal received after being emitted from a transmitter and interacting with clutter (terrain) (12). In this method, the scenario is defined entirely by terrain (SRTM data) and targets (the target model is not discussed in detail), so there is no support for buildings or other large objects which may affect the signal (12). Additionally, there is no support for multipath propagation where the signal travels from transmitter, to target, to ground, to the receiver (or with the target and ground reversed in the order) (12).

1.3.8 Radar Multipath Simulation for Low Altitude Target Above Electrically Large Rough Surface

The method described in (13) for simulation of radar signals for low altitude targets above a moderately rough surface achieves high computational efficiency by constraining the simulation scenario (13). This is effective for a single target over a single, large, moderately rough surface but is not generally-applicable to scenarios that violate these constraints (for example, urban surveillance). The possibility of extending this method to apply it in other environments is mentioned as a possible future work (13).

1.3.9 Synthetic Imagery Generation for Applications in Wide Area Surveillance

In (14) a system is presented for generating synthetic images, aimed at applications in wide area surveillance. The framework presented here also covers some additional aspects of the scenario simulation by simulating vehicle trajectories using SUMO (14). This method achieves a unified framework for simulation of video data for various scenarios, making use of 3D models, including several tools for scenario modeling (14). Only video data can be simulated in this framework, so it cannot be used to simulate active sensors such as LIDAR or Doppler radar.

1.3.10 Efficient Discrete (3D Raster) Ray Tracing

Discrete ray tracing is shown in (15) to be an efficient means of rendering 2D or 3D images from a voxelized model. This has applications in visualization of medical imaging data sets where voxelized models are regularly used, but it is also possible to process geometric models in a voxelization phase before performing discrete ray tracing (15). The method outlined does not account for phase, frequency, or timing information in the rendering process, so it is not directly suitable for simulation of active sensors with Doppler capability.

1.4 Objectives

The proposed method aims to address the gaps in existing methods described in section 1.3, particularly regarding the simulation of active sensor signals in the presence of multipath propagation including transmission of electromagnetic radiation. The proposed method also aims to be general, such that it can be used to simulate signals from different types of active and passive sensors. Finally, the proposed method aims to be computationally feasible. As sensor data simulation is usually an offline process, increasing computation time in order to obtain a more accurate simulation is acceptable, but computation time should not be increased to the point where simulation of a realistic scenario is no longer feasible. The contribution of this paper is to present a computationally-feasible method for combining techniques from computer graphics and active sensor simulation that allows for a unified methodology for simulation of data from multiple different sensor types using 3D models.

Chapter 2

Background

2.1 Physics of Waves

2.1.1 Doppler Effect

When a wave reaches by an observer that is moving relative to the source of the wave the frequency of the wave changes, this is known as the Doppler effect (16). The change in frequency is given by equation 2.1, where Δv is the closing speed of the source and the observer, c is the speed of light, and f_0 is the frequency of the emitted wave (17).

$$\Delta f = \frac{\Delta v}{c} f_0 \tag{2.1}$$

2.1.2 Phase

The phase of a wave is the position of the wave in its cycle. Equation 2.2 is for a sinusoidal wave with phase φ and phase offset φ_0, where f is the frequency of the wave and t is the time. The phase offset refers to the position of the wave in the waveform cycle when $t = 0$.

$$sin(\varphi) = sin(\varphi_0 + 2\pi f t) \tag{2.2}$$

2.2 Active Electromagnetic Sensors

Active electromagnetic sensors (including Doppler Radar and Doppler Lidar) work by transmitting a pulse of electromagnetic radiation from one or more transmitters and measuring the pulse returned to one or more receivers via reflection off of objects in the environment. The returned pulse is used to estimate the positions and radial velocities of objects in the environment.

2.2.1 Transmitted Pulse

When performing a dwell, the active sensor transmitter transmits a pulse with amplitude A, frequency f_p, phase shift φ, and pulse width w_p. Figure 2.1 shows a plot of a transmitted pulse with a sinusoidal carrier wave.

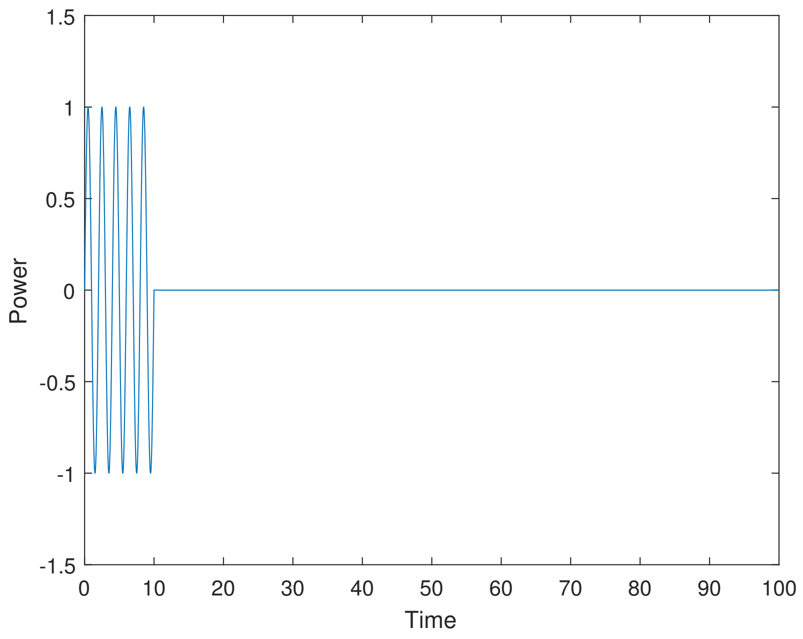

Figure 2.1: A sample signal transmitted by an active electromagnetic sensor transmitter.

2.2.2 Received Pulse

The receiver senses the pulse returned after reflecting from objects in the environment. The pulse received is similar to the transmitted pulse, it is sinusoidal but is phase-shifted, Doppler-shifted, and with a reduced amplitude. If only a single object in the environment reflects the transmitted signal, there will be a single returned pulse. If multiple objects in the environment reflect the transmitted signal there may be multiple pulses returned. Figure 2.2 show a plot of a received pulse with one object in the environment reflecting the transmitted pulse.

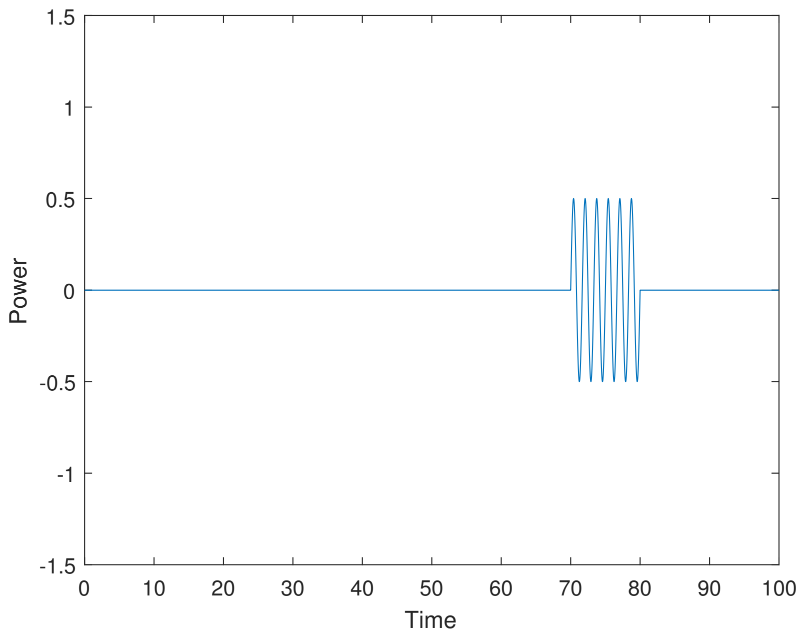

Figure 2.2: A sample signal received by an active electromagnetic sensor receiver.

2.3 Computer Graphics & Rendering

2.3.1 Scenario Modelling

A number of widely-used formats exist for modelling the scene geometry and material properties of 3D environments (18). Additionally, many tools are available for creating and modifying 3D models in common formats, one example that is free and open source is Blender (4). Several models may be composed to form the 3D environment. The scene geometry and material properties are used to determine how rays propagate through the environment during ray tracing (see 2.3.2 for more details).

When selecting a model format for physical simulation it is important that the format supports the material properties that are relevant to the simulation. The proposed simulation method uses the OBJ format with the MTL materials format in its implementation, but other model formats could be used to achieve similar results. It is important to note that the material optical properties are specific to a particular wavelength of electromagnetic radiation (19). For example, different parameters would be required for a radar simulation than for a visible light simulation.

2.3.2 Monte Carlo Path Tracing & Photon Mapping

Monte Carlo path tracing & photon mapping is a two-stage method used to approximate the rendering equation, typically for the purposes of generating computer graphics from 3D models (20; 21). It has been chosen as the basis for the proposed method because the geometric information about illumination paths between the transmitter and the receiver is available during both ray tracing stages and simply needs to be stored (see section 3). Additionally, this method is computationally efficient and allows for an adjustable trade-off between computational load and the accuracy of the simulation (21).

2.3.2.1 The Rendering Equation

The rendering equation is an integral equation for the intensity of light transported from one point to another (22). A form of the rendering equation that is suitable for use in computer graphics was introduced in (22), and can be solved using Monte Carlo methods but this is computationally expensive (20). In (21) an alternative form of the rendering equation is presented by splitting the equation into components based on the type of illumination (equation 2.3). This form of the rendering equation is appropriate for use in photon mapping rendering algorithms and is the form used as the basis for the proposed simulation method.

$$
\begin{aligned}
L_o(x, \overrightarrow{\omega}) = L_e(x, \overrightarrow{\omega}) \\
&+ \int_{\Omega_x} f_r(x, \overrightarrow{\omega}', \overrightarrow{\omega}) L_{i,l}(x, \overrightarrow{\omega}')) cos(\Theta_i) d\omega_i' \\
&+ \int_{\Omega_x} f_{r,s}(x, \overrightarrow{\omega}', \overrightarrow{\omega}) (L_{i,c}(x, \overrightarrow{\omega}') + L_{i,d}(x, \overrightarrow{\omega}')) cos(\Theta_i) d\omega_i' \\
&+ \int_{\Omega_x} f_{r,d}(x, \overrightarrow{\omega}', \overrightarrow{\omega}) L_{i,c}(x, \overrightarrow{\omega}')) cos(\Theta_i) d\omega_i' \\
&+ \int_{\Omega_x} f_{r,d}(x, \overrightarrow{\omega}', \overrightarrow{\omega}) L_{i,d}(x, \overrightarrow{\omega}')) cos(\Theta_i) d\omega_i'
\end{aligned}
\tag{2.3}
$$

2.3.2.2 Approximating the Rendering Equation

In (20; 21) a method for approximating the rendering equation using Monte Carlo path tracing and photon mapping is described in detail. For convenience this method is summarized here, but for a thorough understanding it is recommended to read the original paper. This method traces paths of illumination through the environment from two directions, from the light sources (photon mapping stage) and from the sensor receiver (Monte Carlo path tracing stage) (20; 21). This is done for the sake of computational efficiency as discussed in (20; 21). In each stage, the power of illumination is tracked at each surface interaction as the path of illumination is traced through the environment from the transmitter to the receiver (20; 21). The total illumination power reaching the receiver's position from the direction of a particular pixel is an estimate of the rendering equation and is used to calculate the brightness of that pixel in the rendered image (20; 21).

For the purposes of this paper, it is important to understand that this method involves determining the geometry of sampled illumination paths through the environment as well as the power of illumination along those paths, and that this information is used to approximate the rendering equation. Figures 2.3 and 2.4 show a simplified example of how the geometry is obtained from this process. For a full description of this method refer to (21).

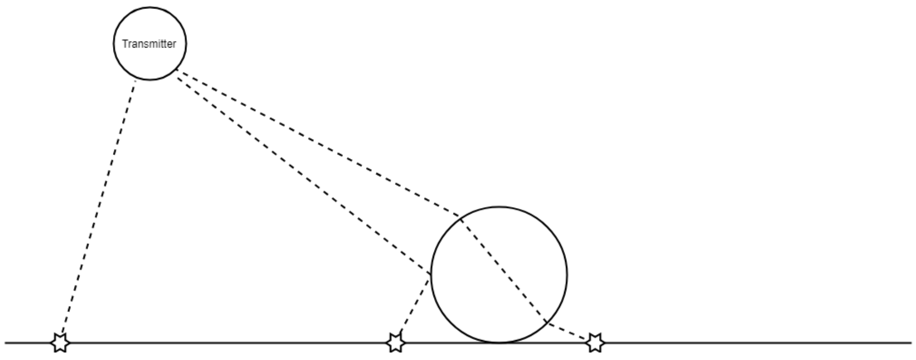

Figure 2.3: The photon mapping stage involves tracing photon rays (shown as dashed lines) from the transmitter, when the photons engage in a diffuse surface interaction a photon is stored at the location of that interaction (each marked with a star). From left to right, the photons shown result in direct diffuse illumination, indirect diffuse illumination via specular reflection, and indirect diffuse illumination via transmission.

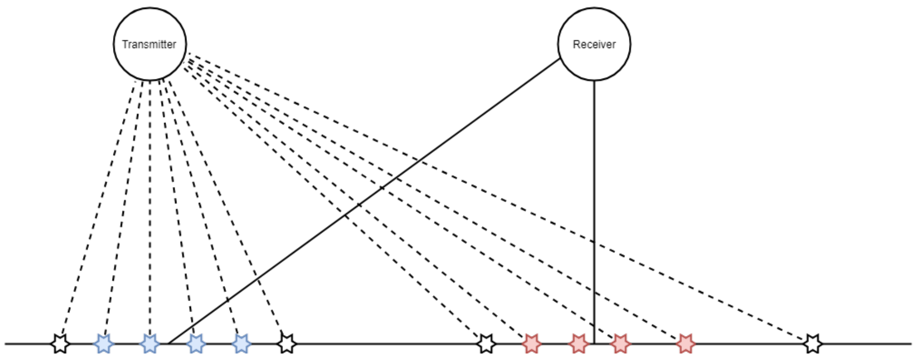

Figure 2.4: In the Monte Carlo path tracing stage sample rays are traced from the receiver, two samples are shown here as solid lines. When the sample rays engage in a diffuse surface interaction, nearby photons are used to estimate the diffuse illumination arriving at the surface point. In this example, 4 photons (coloured) are used to estimate the diffuse illumination.

Chapter 3

Proposed Extension to Existing Methods

3.1 Extended Monte Carlo Path Tracing & Photon Mapping

In the method presented in (21; 20), only information about the intensity of electromagnetic radiation is stored during the rendering process. This is sufficient for standard computer graphics rendering (simulation of visible light images), but it is not sufficient for simulation of active sensor data including timing, frequency, and phase information. To simulate active sensor data, it is also necessary to store information about the path taken by the light as it moves through the environment, including the distance of each leg of the path and the velocity of all surfaces the light has interacted with.

To obtain an unbroken geometric illumination path, photons used to estimate the diffuse illumination are assumed to be located at the point where the diffuse illumination is being estimated. This introduces some error in the path geometry, but when photon density is high the error should be small because the photons are located very near to the point where the diffuse illumination is being estimated. See figures 2.3 and 2.4 for a simplified illustration of the photon mapping and path tracing steps.

3.1.1 Storage of Path History

In standard computer graphics rendering applications, only the illumination intensity is required to compute the pixel intensity for the rendered image. In these methods the illumination intensity can be simply added up along the illumination paths traced in the photon mapping and Monte Carlo path tracing stages. However in order to reconstruct a signal from an active sensor in the time domain more information is required. The illumination intensity must be separated into terms, each term associated with a particular path that the electromagnetic radiation travelled in order to reach the receiver from a transmitter (the illumination path). In addition, for each term the illumination path history must be stored. That is, for each section of the illumination path between surface interactions the location and the velocity of the surface point of the start of that section is recorded in order from transmitter to receiver. The following data structures, described in the C++ programming language, show how this can be accomplished.

```
struct Term
{
    float power;
    std::vector<PathHistory> history;
}

struct PathHistory
{
    float position [3];
    float velocity [3];
}
```

The approximation of the rendering equation that was previously given by a single scalar value for illumination power is now given by an array of terms. This array of terms can be used to reconstruct an IQ signal (as described in section 3.2.2) or to visualize the range-Doppler spectrum of the sensor dwell.

3.1.2 Calculation of Doppler Shift

With the illumination intensity divided into terms, each with a path history, computing the Doppler shift for the term is straightforward. First, the closing speed v_c of the two objects (with positions x_0 and x_1 and moving with velocities v_0 and v_1, respectively) is approximated numerically using equation 3.1, where ϵ is an arbitrarily small time increment. An approximation of the Doppler shift for the wave emitted from one object and observed by the other object is given by equation 3.2, obtained by substituting equation 3.1 into equation 2.1.

$$v_c \approx \frac{|x_0 - x_1| - |x_0 + \epsilon v_0 - x_1 - \epsilon v_1|}{\epsilon} \tag{3.1}$$

$$doppler(x_0, x_1, v_0, v_1, f) = \frac{|x_0 - x_1| - |x_0 + \epsilon v_0 - x_1 - \epsilon v_1|}{\epsilon c} f \tag{3.2}$$

In order to compute the Doppler shift over the course of the entire illumination path the recursive relation given by equation 3.3 can be used. This equation gives the frequency of the electromagnetic radiation along the path after the kth leg, where x_k and v_k are the position and velocity respectively at the start of leg k and $f_{i,j,0}$ is the transmitter frequency. For an illumination path with N legs, the frequency of electromagnetic radiation reaching the receiver is given by $f_{i,j,N}$, abbreviated as $f_{i,j}$ where i is the sample index, j is the term index, and k is the path leg index.

$$f_{i,j,k} = f_{i,j,k-1} + doppler(x_{k-1}, x_k, v_{k-1}, v_k, f_{i,j,k-1}) \tag{3.3}$$

3.1.3 Calculation of Phase Shift

The calculation of the phase shift, given the path history of the term, is also straightforward. Equation 3.4 gives the phase shift for a wave travelling between two points x_0 and x_1 with frequency f.

$$phase(x0, x1, f) = \frac{|x_0 - x_1| 2\pi f}{c} \tag{3.4}$$

The phase shift over the entire illumination path is simply the sum of the phase shifts over each leg of the path, as given by equation 3.5, where i is the sample index, j is the term index, and k is the path leg index.

$$\varphi_{i,j} = \sum_{k=1}^{N} phase(x_{k-1}, x_k, f_{i,j,k-1}) \tag{3.5}$$

3.1.4 Calculation of Pulse Timing

The time of the pulse return over a particular illumination path is depends only on the distance travelled over the course of the path and is given by equation 3.6, where i is the sample index, j is the term index, and k is the path leg index.

$$t_{i,j,r} = \frac{\sum_{k=1}^{N} |x_{k-1} - x_k|}{c} \tag{3.6}$$

The width of the returned pulse will depend on the transmitted pulse width and the frequency of the returned pulse. The number of periods in the returned pulse should be the same as the number of periods in the transmitted pulse. Assuming the transmitter frequency f_t and the transmitted pulse width $w_{i,j,t}$ are given, equation 3.7 yields the number of periods in the pulses, $N_{periods}$. Equation 3.8 gives the width of the returned pulse, $w_{i,j,r}$.

$$N_{periods} = f_t w_{i,j,t} \tag{3.7}$$

$$w_{i,j,r} = \frac{N_{periods}}{f_{i,j}} \tag{3.8}$$

3.2 Example Simulation: Radar IQ Signal

In this example the proposed method was used to simulate a radar IQ signal. Though this was the primary objective, other output formats were also simulated including camera images and range-Doppler spectra which were used for verification and illustrative purposes. The proposed method of extended Monte Carlo path tracing and photon mapping is used throughout, but only the IQ signal simulation utilizes the timing, frequency, and phase information together for a coherent signal reconstruction. The primary sensor data output format is a digitized radar IQ signal for a single dwell. To obtain data for a sweep, multiple dwells can be simulated independently. The models used for all scenarios discussed below (targets, urban setting, and the radar transmitters) are Wavefront OBJ models with MTL material library files.

3.2.1 Sampling in Antenna Pattern

The antenna pattern assumed for this simulation is a simple cone pattern with uniform vertical and horizontal radiation patterns. Therefor the sampling pattern for the rendering equation has all rays originating from the sensor position, with the directions of each ray uniformly distributed within a conical beam with a given vertex angle. A greater number of samples will yield a better approximation of the returned radar signal.

It should be noted that a conical beam pattern was chosen for the sake of simplicity in terms of interpreting and evaluating the results of the proposed simulation method. Realistic antenna patterns are not perfectly uniform and conical, and have a variety of radiation patterns depending on the application (23; 24; 25). It is possible to use importance sampling to weight the returned signal samples according to an arbitrary antenna radiation pattern. Importance sampling can also be used during other stages of rendering to mimic the radiation pattern transmitted by the radar transmitter.

3.2.2 IQ Signal Construction

In standard computer graphics rendering the rendering equation for each pixel is approximated by summing the scalar terms from the sample(s) corresponding to that pixel. To reconstruct the radar IQ signal, sinusoidal pulse terms from all samples within the beam pattern are summed.

$$rect(t, t_0, w) = \begin{cases} 1 & t_0 \leq t \leq t_0 + w \\ 0 & else \end{cases} \tag{3.9}$$

$$I(t) = \sum_{i=1}^{N_{samples}} \sum_{j=1}^{N_{terms}} rect(t, t_{i,j,r}, w_{i,j,r}) A_{i,j} cos(\varphi_{i,j} + 2\pi f_{i,j}(t - t_{i,j,r})) \tag{3.10}$$

$$Q(t) = \sum_{i=1}^{N_{samples}} \sum_{j=1}^{N_{terms}} rect(t, t_{i,j,r}, w_{i,j,r}) A_{i,j} sin(\varphi_{i,j} + 2\pi f_{i,j}(t - t_{i,j,r})) \quad (3.11)$$

To reduce memory consumption the in-phase and quadrature components of the signal are quantized during the signal construction process. Equations 3.12 and 3.13 give the quantized signal components, where T_s is the sampling period. The sampling period should be chosen such that the signal can be reconstructed with sufficient accuracy.

$$I[k] = I(kT_s) \quad (3.12)$$

$$Q[k] = Q(kT_s) \quad (3.13)$$

3.2.3 Receiver Noise

The signal reconstructed in section 3.2.2 is an approximation of the pure signal returning from the environment and does not account for noise introduced by the radar receiver. This noise can be modelled by additive zero-mean Gaussian white noise (26). Equations 3.14 and 3.15 show the quantized signal with noise, where the variance σ^2 corresponds to the noise power of the receiver.

$$I'[k] = I[k] + \mathcal{N}(0, \sigma^2) \quad (3.14)$$

$$Q'[k] = Q[k] + \mathcal{N}(0, \sigma^2) \quad (3.15)$$

3.2.4 Results

This section contains output generated by using the proposed method. Several simplified scenarios were simulated for verification purposes. A more complex, realistic scenario was also simulated. For each scenario a camera image, a radar range-Doppler spectrum, and a radar IQ signal (with and without receiver noise) were simulated. The camera image is included to provide a visual reference to the layout of each scenario.

3.2.4.1 A Simplified Scenario with Diffuse Illumination

This scenario consisted of a single spherical target in the center of the field of view of the sensor, and moving directly towards the sensor at a constant velocity. The target was made of a material that is entirely diffuse, and since there was only was object in the scene all illumination is direct diffuse illumination. For all sensor types the sensor was stationary. The results are shown in figures 3.1 through 3.6.

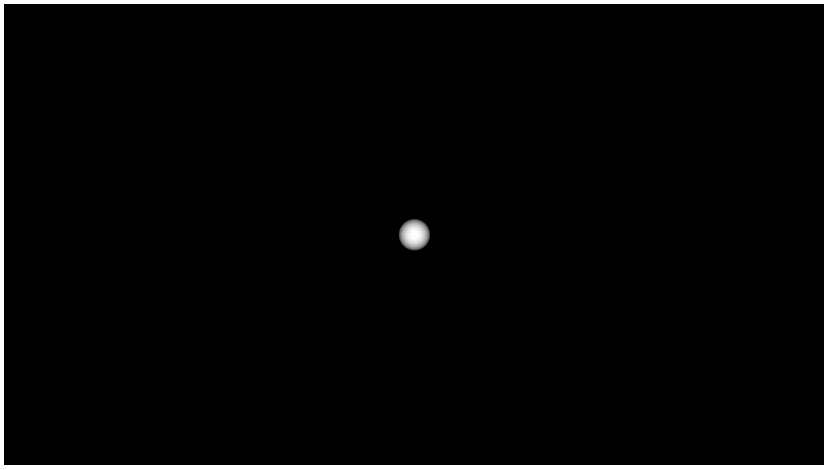

Figure 3.1: A simulated camera image for the scenario described in section 3.2.4.1. The spherical target is visible in the center of the image and is illuminated from the direction of the sensor.

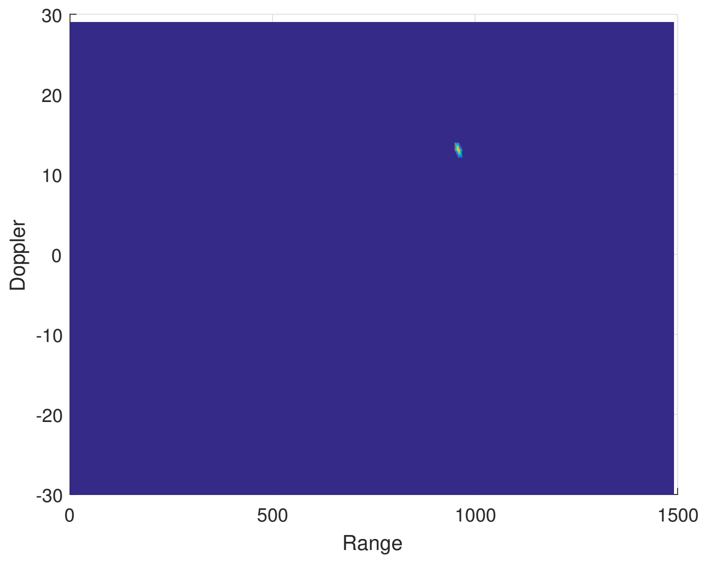

Figure 3.2: A simulated range-Doppler spectrum for the scenario described in section 3.2.4.1. A peak is visible where the signal was returned by the spherical target.

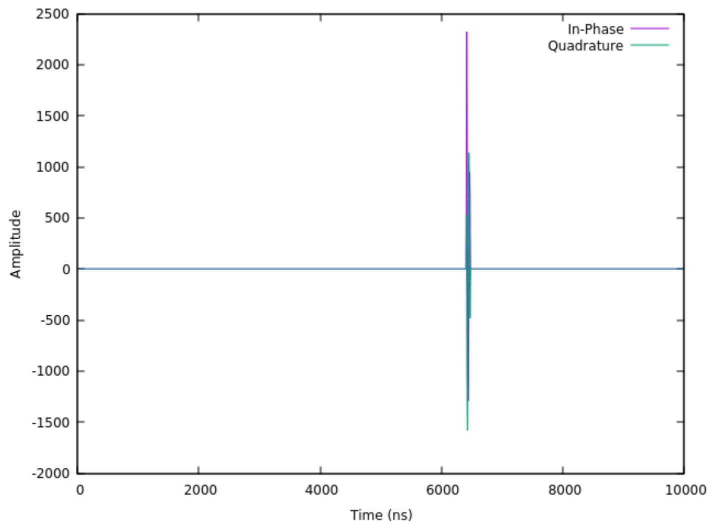

Figure 3.3: A simulated radar IQ signal for the scenario described in section 3.2.4.1. A peak is visible where the signal was returned by the spherical target.

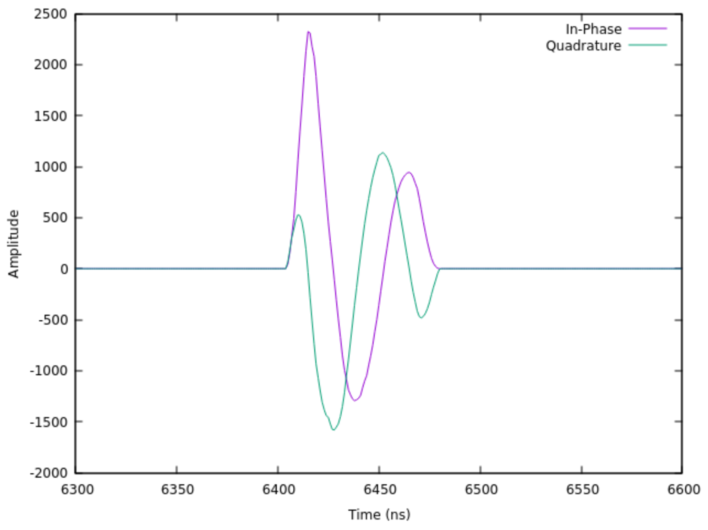

Figure 3.4: A simulated radar IQ signal for the scenario described in section 3.2.4.1. This figure shows a more detailed view of the peak returned by the spherical target.

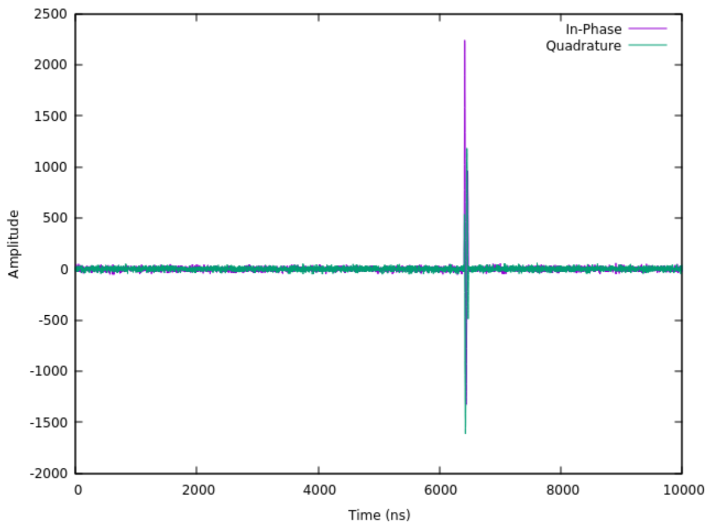

Figure 3.5: A simulated radar IQ signal for the scenario described in section 3.2.4.1. A peak is visible where the signal was returned by the spherical target. A small amount of receiver noise has been added to the signal.

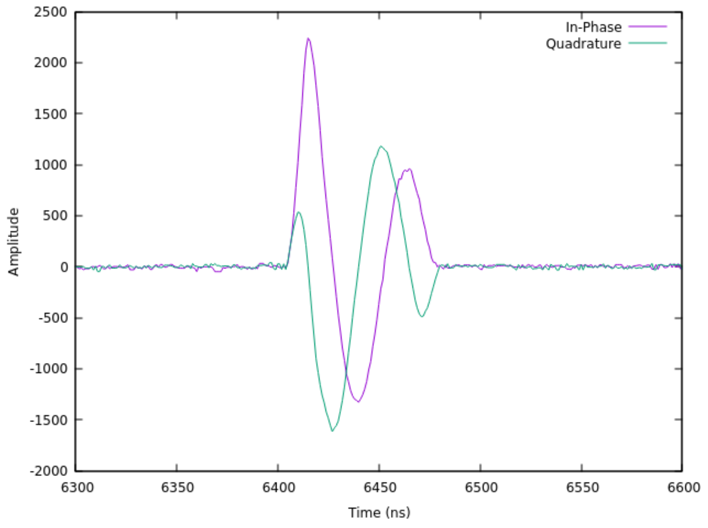

Figure 3.6: A simulated radar IQ signal for the scenario described in section 3.2.4.1. This figure shows a more detailed view of the peak returned by the spherical target. A small amount of receiver noise has been added to the signal.

3.2.4.2 A Simplified Scenario with Transmission of Illumination

This scenario was similar to the scenario described in section 3.2.4.1, but a stationary pane was added between the sensor and the spherical target. The pane was made of a material that is partially diffuse but also partially transmissive. This allowed for the spherical target to be indirectly illuminated by electromagnetic radiation transmitted through the pane. Support for these effects was one of the objectives for the proposed method. The results are shown in figures 3.7 through 3.12.

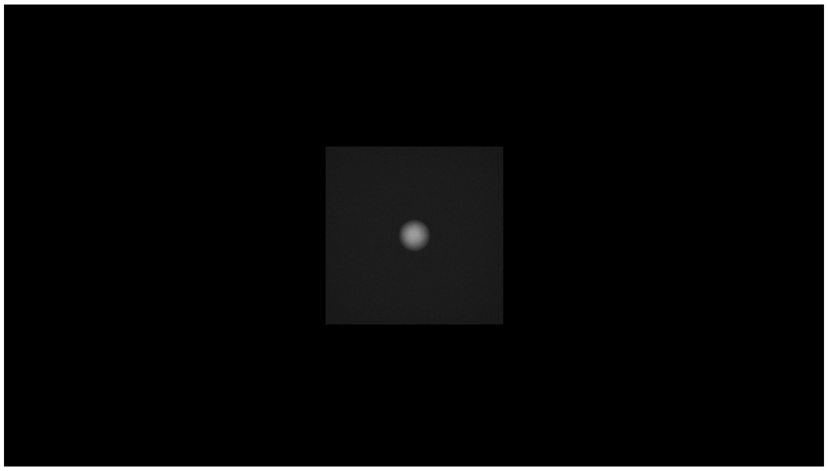

Figure 3.7: A simulated camera image for the scenario described in section 3.2.4.2. The spherical target is visible in the center of the image, seen through the transmissive pane. The transmissive pane is also partially visible because of its material's diffuse component.

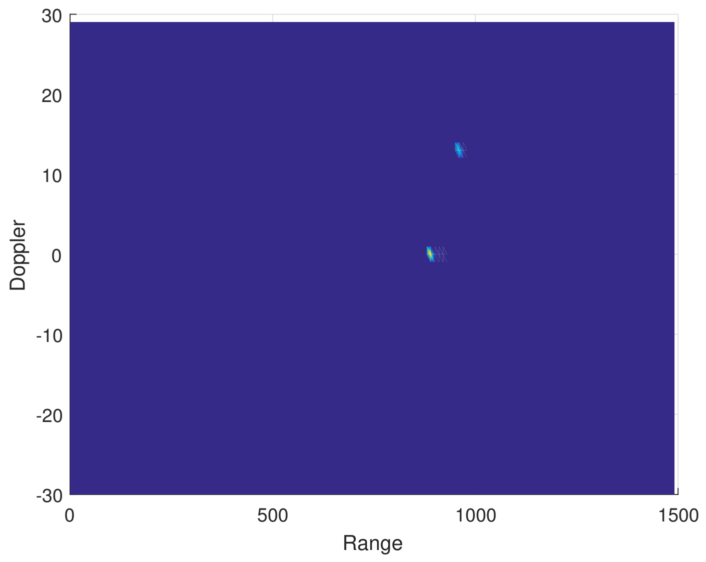

Figure 3.8: A simulated range-Doppler spectrum for the scenario described in section 3.2.4.2. Peaks are visible where the signal was returned by the spherical target and the pane.

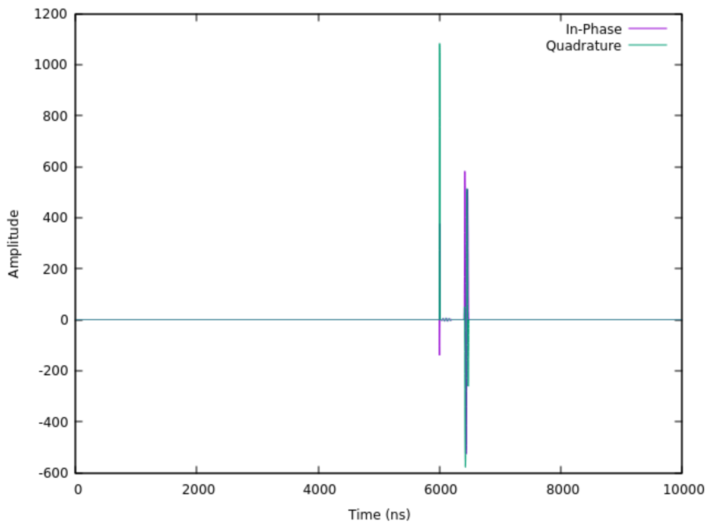

Figure 3.9: A simulated radar IQ signal for the scenario described in section 3.2.4.2. Peaks are visible where the signal was returned by the spherical target and the pane.

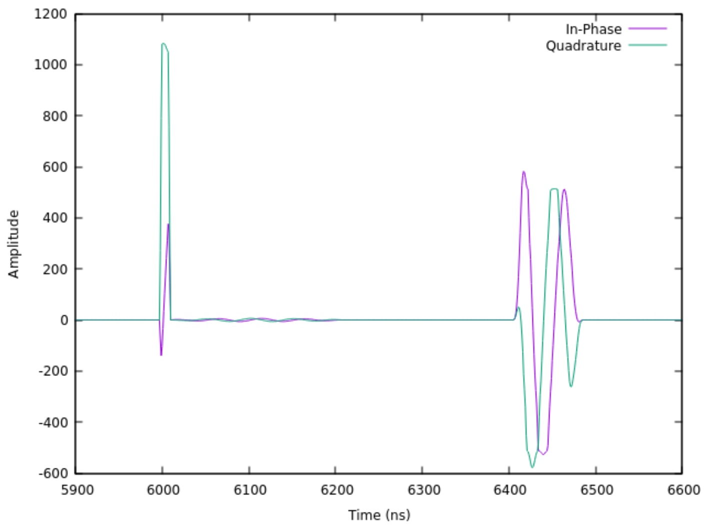

Figure 3.10: A simulated radar IQ signal for the scenario described in section 3.2.4.2. This figure shows a more detailed view of the area where the peaks returned by the spherical target and the pane are.

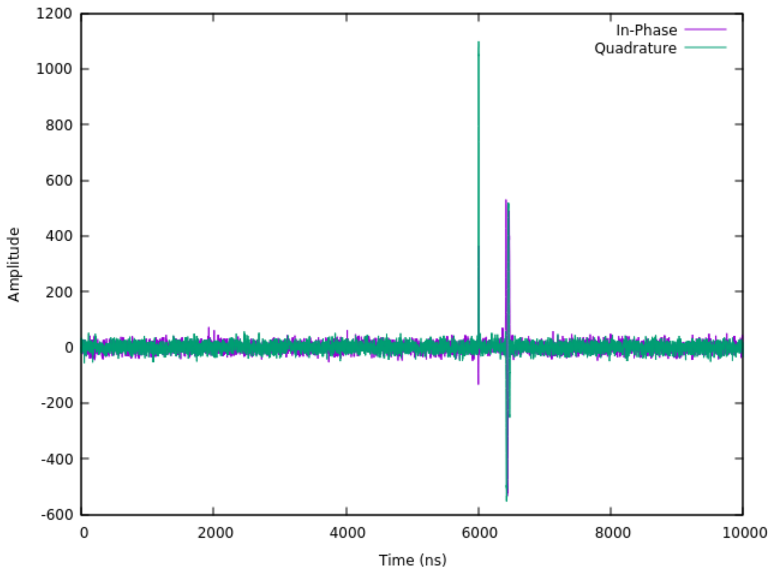

Figure 3.11: A simulated radar IQ signal for the scenario described in section 3.2.4.2. Peaks are visible where the signal was returned by the spherical target and the pane. A small amount of receiver noise has been added to the signal.

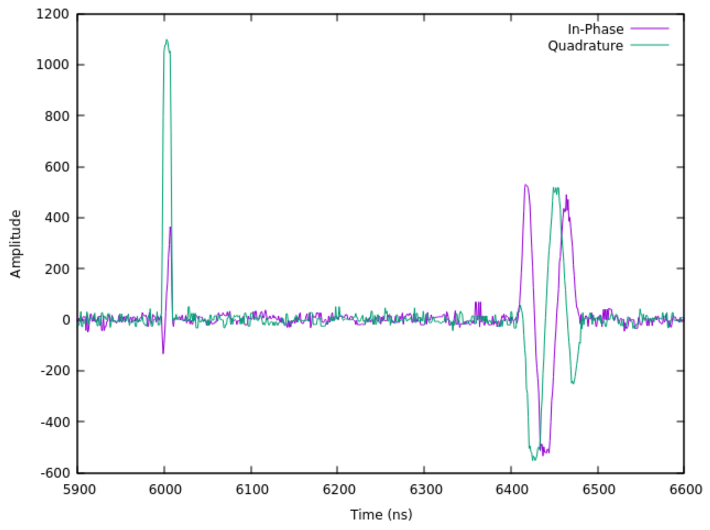

Figure 3.12: A simulated radar IQ signal for the scenario described in section 3.2.4.2. This figure shows a more detailed view of the area where the peaks returned by the spherical target and the pane are. A small amount of receiver noise has been added to the signal.

3.2.4.3 A Realistic Scenario with Multiple Targets and Clutter

This scenario was simulated to model a more realistic use case. In this example an air-to-ground surveillance scenario was modelled with multiple targets (ground vehicles) moving with constant velocity in an urban environment. The sensors were also moving with a constant velocity.

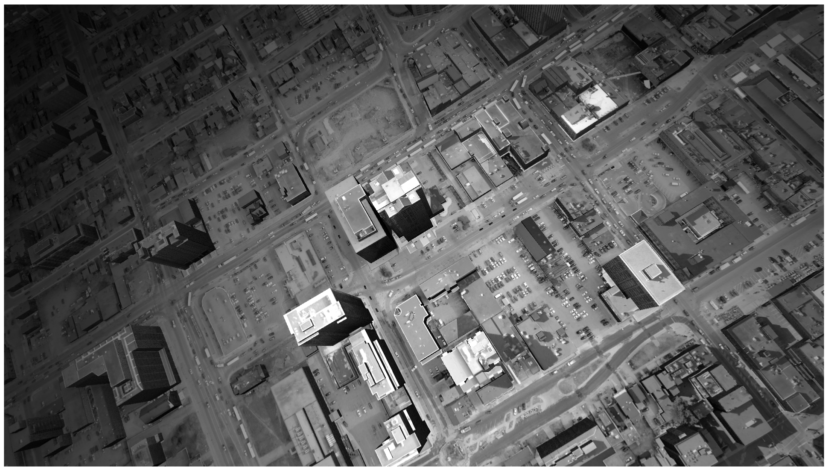

Figure 3.13: A simulated camera image for the scenario described in section 3.2.4.3. Close inspection reveals multiple targets traveling in the urban environment.

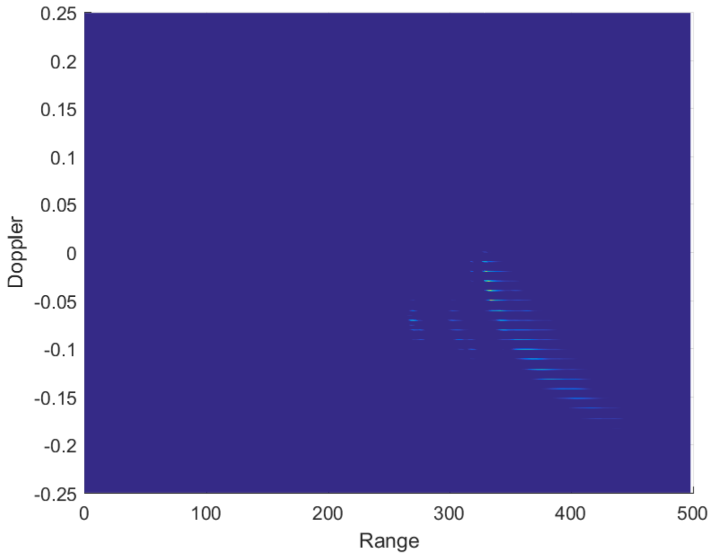

Figure 3.14: A simulated range-Doppler spectrum for the scenario described in section 3.2.4.3. Multiple peaks at various range-Doppler combinations are visible.

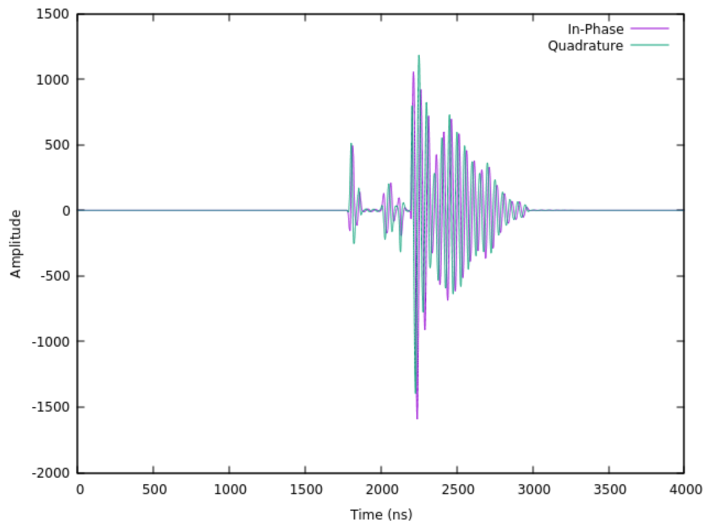

Figure 3.15: A simulated radar IQ signal for the scenario described in section 3.2.4.3. Electromagnetic radiation is returned over multiple paths (multipath propagation) and the resulting signal components interfere with one another in the returned signal.

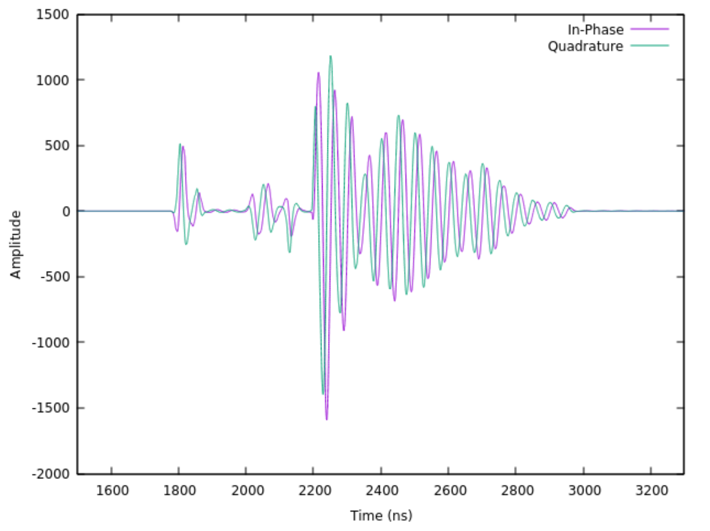

Figure 3.16: A simulated radar IQ signal for the scenario described in section 3.2.4.3. A more detailed view of the signal returned.

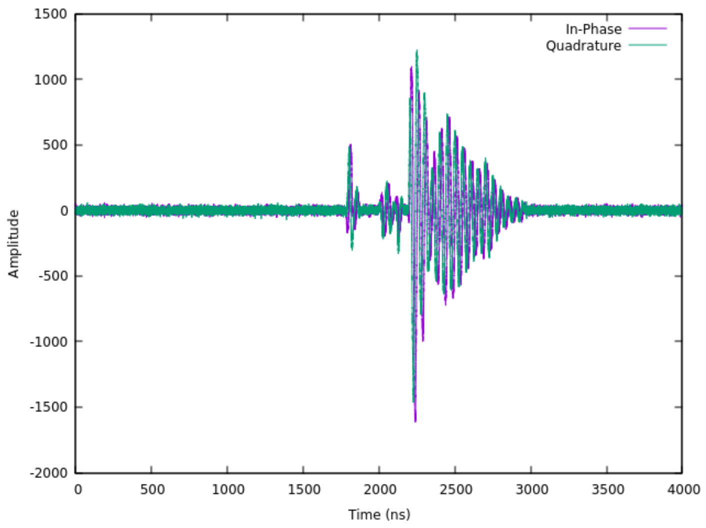

Figure 3.17: A simulated radar IQ signal for the scenario described in section 3.2.4.3. Electromagnetic radiation is returned over multiple paths (multipath propagation) and the resulting signal components interfere with one another in the returned signal. A small amount of receiver noise has been added to the signal.

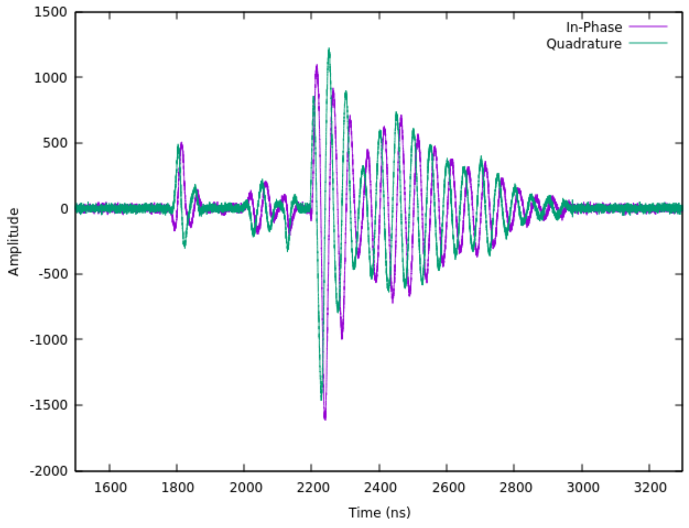

Figure 3.18: A simulated radar IQ signal for the scenario described in section 3.2.4.3. A more detailed view of the signal returned. A small amount of receiver noise has been added to the signal.

3.2.5 Comparison with Simulation Using the Blender Renderer

The Blender renderer simulation (see section 1.3.1) was chosen for comparison with the proposed method due to the similarity of their objectives and output. Both methods aim to simulate radar data using 3D models to construct a scenario. Both methods also use ray tracing to accomplish this, and each offers some level of support for simulation of multipath propagation effects (though to different extents). There are also some differences between the methods. The Blender renderer method aims to be a simple solution that leverages an existing ray tracer to achieve approximated results (4). This differs from the proposed method, which instead aims to be detailed and accurate.

Simulation using the Blender renderer has two main limitations (4). The first limitation is with signal penetration, and the second is with multipath propagation (4). Both of these limitations effectively boil down to the same issue, that the information provided by the Blender renderer does not allow for correct modeling of range or frequency information in these cases. The Blender Z-pass and Vector pass provide distance and velocity information for each pixel, but only account for the distance and velocity of the nearest object in the direction of each pixel (5). When considering signal penetration (transmission) and multipath propagation this information is insufficient to correctly reconstruct the returned signal. Incoming electromagnetic radiation from a particular direction (corresponding to a pixel) may be from different sources and have different frequency components with different timing and phase.

Figures 3.19, 3.20, and 3.21 show data simulated using the Blender renderer method. These figures correspond to the scenarios described in sections 3.2.4.1, 3.2.4.2, and 3.2.4.3, respectively. These simulated scenarios will be used for comparison of the two methods. Note that no phase information is available when using the Blender renderer method. Because of this a range-Doppler spectrum can be produced but the IQ signal cannot be calculated.

When comparing the two methods for the scenario described in 3.2.4.1, the results are the same (see figures 3.19 and 3.2). This is due to the fact that the scenario being simulated includes only direct illumination for a monostatic radar setup (in which the transmitter and receiver are at the same location (27)). In this case, nothing is lost by using the simplified range and velocity modeling considering only the nearest object. Note that even with only direct illumination, this is only true for a monostatic radar setup. For a multistatic setup (in which there are one or more transmitters at different locations than the receiver (21)) the Blender renderer method is not reliable because only the distance between the nearest object and the receiver is considered (5).

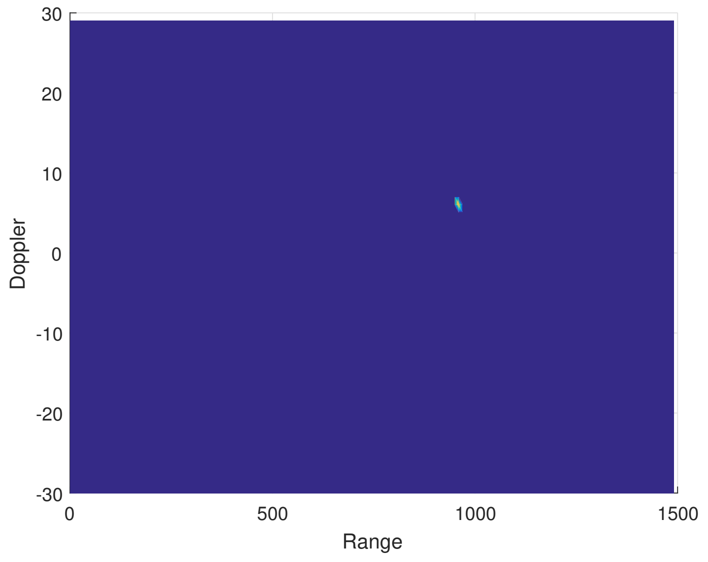

Figure 3.19: A simulated range-Doppler spectrum for the scenario described in section 3.2.4.1, generated using the Blender renderer method. For this scenario the results are correct because there is only direct diffuse illumination.

The limitations of the Blender renderer method are apparent when considering the scenario described in section 3.2.4.2. Figure 3.20 illustrates the incorrect modeling of range and doppler information for signal penetration (which is a special case of multipath propagation) in the Blender renderer method. The signal returned from the more distant object after penetration of the transmissive pane is erroneously included in the range bin at the distance of the pane. The Doppler bin of the signal returned from the spherical object is also incorrect, since the object is in motion but the velocity registered is from nearest object (the pane), which is stationary. The proposed method correctly models these effects, as seen in figure 3.8.

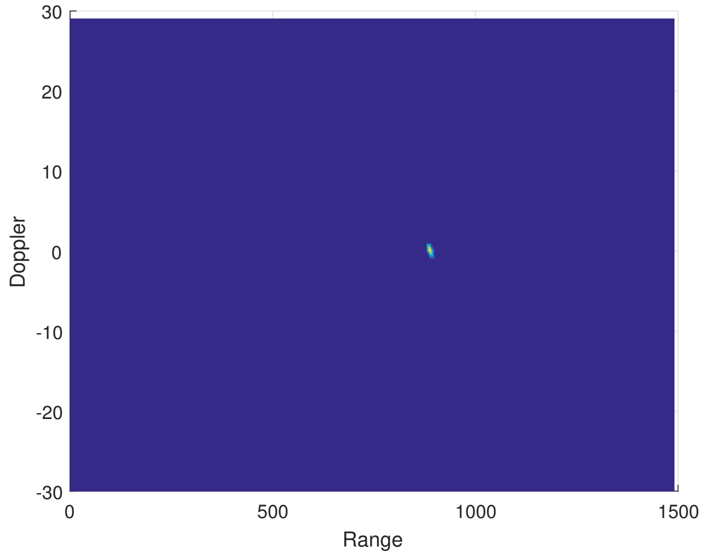

Figure 3.20: A simulated range-Doppler spectrum for the scenario described in section 3.2.4.2, generated using the Blender renderer method. A single peak is visible where the signal was returned by the spherical target and the pane. The total signal amplitude returned is correct, but it is not correctly distributed into range and Doppler bins. The correctly modeled results can been seen in figure 3.8.

A direct comparison of specific signal returns is more difficult to compare for the more realistic scenario described in section 3.2.4.3 due to the increased complexity. Nevertheless, comparing figures 3.14 (simulated by the proposed method) and 3.21 (simulated using the Blender renderer method) gleans some insight. The range-Doppler spectrum generated using the proposed method contains additional peaks at larger range and absolute Doppler shift values when compared to the spectrum generated by the Blender renderer method. This is consistent with expectations, since the Blender renderer method will ignore range and Doppler effects from objects during multipath propagation if they are not the nearest to the receiver along the direction sampled. Signal returned along these paths will include the correct amplitude information but will not be correctly binned in terms of range and Doppler.

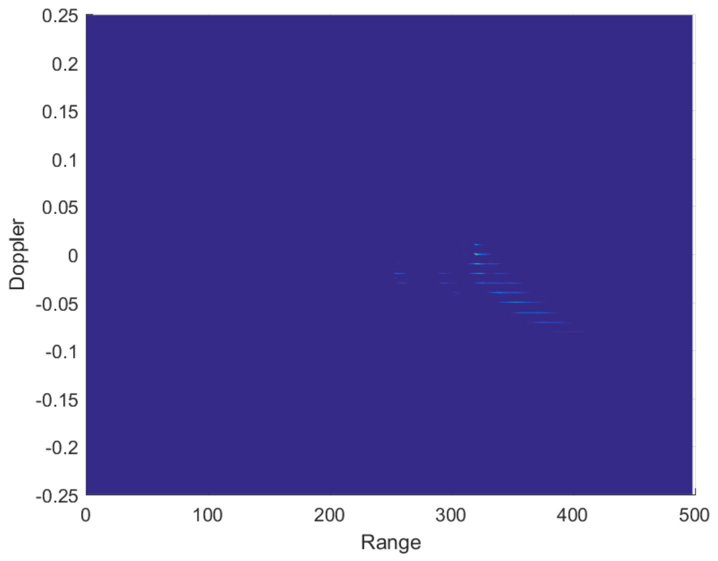

Figure 3.21: A simulated range-Doppler spectrum for the scenario described in section 3.2.4.3, generated using the Blender renderer method. When comparing these results to the results from the proposed method (see figure 3.14) it is apparent that some of the signal returns are missing or incorrectly binned. This is due to the incorrect modeling of range and Doppler information for illumination paths involving multipath propagation.

Table 3.1 summarizes this comparison and describes how each method handles various effects of interest.

	Blender method	Proposed method
Direct illumination	Correct	Correct
Multipath propagation (amplitude)	Correct	Correct
Multipath propagation (range)	Incorrect	Correct
Multipath propagation (Doppler)	Incorrect	Correct
Phase	Not modelled	Correct
Multistatic configuration	Incorrect	Correct

Table 3.1: A summary of a comparison between the Blender renderer method and the proposed method.

Chapter 4

Cloud Computing & Horizontal Scaling

The computation time required to simulate signal-level sensor data using the proposed method can be large depending on the scenario being simulated, the number of frames being simulated, and the required level of accuracy. Simulation of many frames of accurate sensor data for a large-scale scenario can be achieved in a reasonable amount of time by leveraging cloud computing. This section describes one possible distributed system design and evaluates its potential to operate at a large scale.

4.1 Frame Independence

The proposed method generates a single sensor data frame (or dwell), which is generated independently from other frames. Because there is no data dependence between frames this is a suitable level of granularity to divide the workload for the purposes distributed computing. Each frame can be computed on a particular node and the only data that must be transferred is the completed frame data.

4.2 Distributed System Architecture

A grid computing solution was implemented using Sun Grid Engine and computational resources provided by Amazon Web Services (AWS). The distributed system architecture and the hardware configurations are described in this section.

4.2.1 Sun Grid Engine

Sun Grid Engine is a grid computing platform which facilitates the configuration of a network (or grid) of computation nodes to run independent jobs (28). One node in the network is designated as the master node, where jobs are submitted and scheduled to be run to worker nodes (28). The master node can assign jobs to itself, so it still functions as a worker node. For the proposed simulation method, frames can take a relatively long time to compute when compared with the time it takes to schedule jobs, so the performance of master node is not significantly impaired by its additional scheduling responsibilities. Since each frame to be simulated can be considered a job in the context of grid computing, Sun Grid Engine can be easily used to distribute the task of simulating many frames over a grid of computation nodes. All nodes share access to a single network storage node, a process facilitated by Sun Grid Engine. This network storage node is used to host 3D models, parameter files and the software required to run the simulations, as well as to store output data as it is generated. Figure 4.1 shows how the system components are connected.

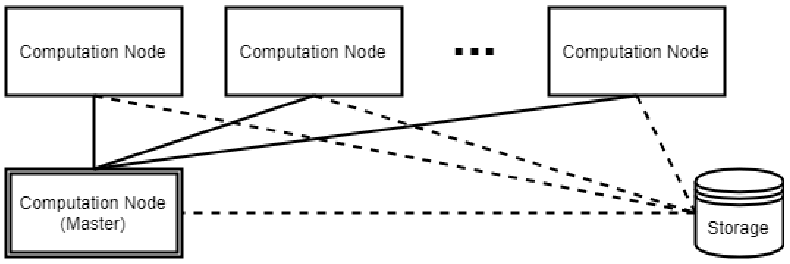

Figure 4.1: The grid computing architecture enabled by Sun Grid Engine consists of a master computation node, a storage node, and (optionally) several additional computation nodes. All nodes have access to the storage node. The master node has access to each of the additional computation nodes and can schedule jobs on these nodes.

4.2.2 Computation Nodes

The computation nodes used for this implementation were AWS Elastic Compute Cloud (EC2) nodes. Table 4.1 describes the hardware of the computation nodes as provided in (29). AWS does not provide full hardware details of its EC2 nodes (29).

EC2 node type	c3.2xlarge
Processor architecture	64-bit
vCPU	8
Memory	15 GB
Local storage	Solid state
Network performance	High

Table 4.1: Computation node details

4.2.3 Storage Node

The storage node used for this implementation was an Amazon Elastic Block Storage (EBS) node. Required files (3D models, parameter files, and the simulation software) were manually added to the storage node during setup. Sun Grid Engine facilitates mounting the storage node to the computation nodes, allowing them access to the required files to run the simulation, and providing storage for their output data.

4.3 Horizontal Scalability Analysis

Run time measurements were taken for grids with 1, 4, and 8 computation nodes. For each grid size, run times were measured with 1, 4, 8, and 16 frames per node, and averaged over 3 runs for each configuration. Tables 4.2, 4.3, and 4.4 summarize the results of these timed runs.

Frames per node	Average run time	Average run time per frame
1	346.33	346.33
4	1356.00	339.00
8	2775.33	346.92
16	5462.00	341.38

Table 4.2: Run time measurements using 1 computation node.

Frames per node	Average run time	Average run time per frame
1	348.25	87.06
4	1394.00	87.13
8	2784.50	87.02
16	5450.00	85.16

Table 4.3: Run time measurements using 4 computation nodes.

Frames per node	Average run time	Average run time per frame
1	331.92	41.49
4	1378.33	43.07
8	2757.67	43.09
16	5557.79	43.42

Table 4.4: Run time measurements using 8 computation nodes.

From tables 4.2, 4.3, and 4.4 it is evident that average run time per frame is approximately constant for a given grid size (number of computation nodes). Table 4.5 shows the overall average run time per frame for each grid size, taking into account all runs regardless of the number of frames per node, as well as the speed up achieved. Figure 4.2 shows a plot of the speedup data, including a trend line. From the trend line we can see that the speed up is approximately equal to the number of nodes in the grid therefor linear speedup has been achieved.

Grid size	Average run time per frame	Speed up
1	343.41	1.00
4	86.59	3.97
8	42.77	8.03

Table 4.5: Average run time measured and speed up for each grid size (number of computation nodes) for any number of frames per node.

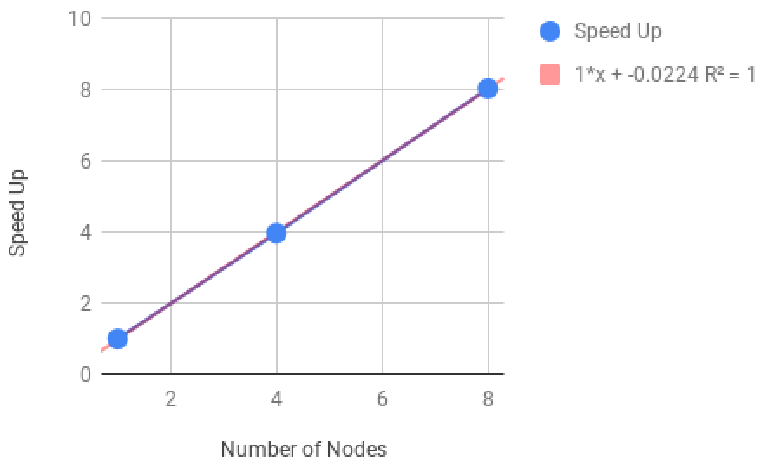

Figure 4.2: Speed up achieved with each grid size (number of computation nodes).

Chapter 5

Conclusion

5.1 Evaluation of Results

The proposed method was successful in achieving a detailed simulation of signal-level sensor data for active and passive sensors. When compared to a similar existing method, the proposed method directly addresses several limitations. Particularly, the ability to accurately model behaviour in scenarios involving multipath propagation of electromagnetic radiation including transmission. This was achieved by recording additional information during the photon mapping and path tracing stages when compared to a standard rendering algorithm.

5.2 Limitations

5.2.1 3D Model Material Properties for Simulation of Various Wavelengths

For realistic simulations a challenge remains in obtaining 3D models which have material parameters configured for various wavelengths of electromagnetic radiation. Many models are widely available for ground vehicles, aircraft, etc. but their material parameters are primarily modeled for visible-spectrum light with parameters specified for red, blue, and green wavelengths. In order to accurately model a scenario involving various sensors operating at different wavelengths, the models' material parameters must be specified for each of the wavelengths being considered. The Wavefront OBJ model format with MTL material library files does have support for defining materials using a spectral curve (30), but models packaged with these files are not readily available.

5.2.2 Computation Time

For complex scenarios covering a large area or with many objects, where a high degree of accuracy is required, or where the simulation length is long, computation time remains a challenge. Horizontal scalability offers one way of dealing with this as discussed in section 4 but can become expensive for a particularly demanding simulation. Reducing the computational load may be possible by further optimizing the rendering process, which could be inspired by existing work in computer graphics.

Bibliography

[1] T. Kirubarajan, Y. Bar-Shalom, K. R. Pattipati, and I. Kadar, "Ground target tracking with variable structure imm estimator," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 36, no. 1, pp. 26–46, 2000.

[2] F. Bashir and F. Porikli, "Performance evaluation of object detection and tracking systems," in *Proceedings 9th IEEE International Workshop on PETS*, 2006, pp. 7–14.

[3] M. Ouza, M. Ulrich, and B. Yang, "A simple radar simulation tool for 3d objects based on blender," in *Radar Symposium (IRS), 2017 18th International.* IEEE, 2017, pp. 1–10.

[4] Blender Foundation. (n.d.) Blender. [Online]. Available: https://www.blender.org/

[5] ——. (n.d.) Render passes - blender manual. [Online]. Available: https://docs.blender.org/manual/en/dev/render/blender_render/settings/passes.html

[6] S. D. Brown, D. D. Blevins, and J. R. Schott, "Time-gated topographic lidar scene simulation," in *Laser Radar Technology and Applications X*, vol. 5791. International Society for Optics and Photonics, 2005, pp. 342–354.

[7] D. Gubelli, O. A. Krasnov, and O. Yarovyi, "Ray-tracing simulator for radar signals propagation in radar networks," in *Radar Conference (EuRAD), 2013 European*. IEEE, 2013, pp. 73–76.

[8] D. W. Hess, "Introduction to rcs measurements," in *Antennas and Propagation Conference, 2008. LAPC 2008. Loughborough*. IEEE, 2008, pp. 37–44.

[9] S. Auer, R. Bamler, and P. Reinartz, "Raysar-3d sar simulator: Now open source," in *Geoscience and Remote Sensing Symposium (IGARSS), 2016 IEEE International*. IEEE, 2016, pp. 6730–6733.

[10] H. R. Anderson, "A ray-tracing propagation model for digital broadcast systems in urban areas," *IEEE Transactions on Broadcasting*, vol. 39, no. 3, pp. 309–317, 1993.

[11] M. Brooker and M. Inggs, "A signal level simulator for multistatic and netted radar systems," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, no. 1, pp. 178–186, 2011.

[12] A. Sudol, S. E. Gordon, H. Ford, E. Inclan, D. N. Mavris, and M. Z. Miller, "Simulation of radar signal propagation via multipath," in *2018 Modeling and Simulation Technologies Conference*, 2018, p. 4060.

[13] P. Peng, L. x. Guo, and C. Tong, "An em model for radar multipath simulation and hrrp analysis of low altitude target above electrically large composite scale rough surface," *Electromagnetics*, vol. 38, no. 3, pp. 177–188, 2018.

[14] E. J. Griffith, C. Mishra, J. F. Ralph, and S. Maskell, “A system for the generation of synthetic wide area aerial surveillance imagery,” *Simulation Modelling Practice and Theory*, vol. 84, pp. 286–308, 2018.

[15] R. Yagel, D. Cohen, and A. Kaufman, “Discrete ray tracing,” *IEEE Computer graphics and applications*, no. 5, pp. 19–28, 1992.

[16] N. Giordano, *College Physics: Reasoning and Relationships.* Cengage Learning, 2009. [Online]. Available: https://books.google.ca/books?id=BwistUlpZ7cC

[17] J. Rosen and L. Gothard, *Encyclopedia of Physical Science*, ser. Facts on File Science Library. Facts On File, Incorporated, 2009. [Online]. Available: https://books.google.ca/books?id=avyQ64LIJa0C

[18] K. McHenry and P. Bajcsy, “An overview of 3d data content, file formats and viewers,” *National Center for Supercomputing Applications*, vol. 1205, p. 22, 2008.

[19] K. Coulson, G. Bouricius, and E. Gray, “Optical reflection properties of natural surfaces,” *Journal of Geophysical Research*, vol. 70, no. 18, pp. 4601–4611, 1965.

[20] H. W. Jensen, “Global illumination using photon maps,” in *Rendering Techniques 96*. Springer, 1996, pp. 21–30.

[21] P. H. Christensen, H. W. Jensen, and F. Suykens, “A practical guide to global illumination using photon mapping,” in *SIGGRAPH Course Notes*, vol. 38, 2001, pp. 60–90.

[22] J. T. Kajiya, “The rendering equation,” in *ACM Siggraph Computer Graphics*, vol. 20, no. 4. ACM, 1986, pp. 143–150.

[23] C. A. Balanis, "Antenna theory: A review," *Proceedings of the IEEE*, vol. 80, no. 1, pp. 7–23, 1992.

[24] B. Carlson, L. Goodman, J. Austin, M. Ganz, and L. O. Upton, "An ultralow-sidelobe adaptive array antenna," in *The Lincoln Laboratory Journal*. Citeseer, 1990.

[25] M. J. Vaughan, K. Y. Hur, and R. C. Compton, "Improvement of microstrip patch antenna radiation patterns," *IEEE transactions on antennas and propagation*, vol. 42, no. 6, pp. 882–885, 1994.

[26] A. W. Doerry, "Noise and noise figure for radar receivers," *Sandia Nat. Lab., Livermore, CA, USA, Tech. Rep. SAND2016-9649*, 2016.

[27] V. S. Chernyak, *Fundamentals of multisite radar systems: multistatic radars and multistatic radar systems*. Routledge, 2018.

[28] G. Borges, M. David, J. Gomes, C. Fernandez, J. Lopez Cacheiro, P. Rey Mayo, A. Simon Garcia, D. Kant, and K. Sephton, "Sun grid engine, a new scheduler for egee middleware," in *IBERGRID–Iberian Grid Infrastructure Conference*. Citeseer, 2007.

[29] Amazon. (n.d.) Previous generation instances. [Online]. Available: https://aws.amazon.com/ec2/previous-generation/

[30] D. Ramey, L. R. Rose, and L. Tyerman. (1995) Mtl obj materials file. [Online]. Available: http://paulbourke.net/dataformats/mtl/