ARL-TR-9841 • Nov 2023

DEVCOM ARMY RESEARCH LABORATORY

Modeling Ultraviolet Wireless Networks in Realistic Environments

by John Reko, Fikadu T Dagefu, Terrence J Moore, Justin Kong, and Hakan Arslan

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-9841 ● Nov 2023

Modeling Ultraviolet Wireless Networks in Realistic Environments

by John Reko
Rice University

Fikadu T Dagefu, Terrence J Moore, and Justin Kong
Army Research Directorate, DEVCOM Army Research Laboratory

Hakan Arslan
Oak Ridge Associated Universities (ORAU)

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
November 2023	Technical Report	May 2023 - August 2023

4. TITLE AND SUBTITLE
Modeling Ultraviolet Wireless Networks in Realistic Environments

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
John Reko, Fikadu T Dagefu, Terrence J Moore, Justin Kong, and Hakan Arslan

5d. PROJECT NUMBER
AH80

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
DEVCOM Army Research Laboratory
ATTN: FCDD-RLA-NB
2800 Powder Mill Road
Adelphi, MD 20783-1138

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-9841

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
primary author's email: <johnreko0808@gmail.com>.
ORCID ID: Terrence J Moore, 0000-0003-3279-296

14. ABSTRACT
With high atmospheric scattering and reduced solar noise, the ultraviolet-C (UV-C) band has been investigated as a short-range, secure, non-line-of-sight optical communication system. We created software that allows the user to input a set of UV-C transceiver locations, physical obstructions, and network parameters to generate a UV-C network. After network generation, the user can create matrices that contain information about the optimal steering angles, bit error rate, and maximum data rate for every communication link in the network. The matrices are generated according to single scattering UV-C communication models and assuming an on-off-keying modulation scheme. Additionally, a 4D array is generated containing information about the interference of two pairs of transceivers. This simulation data can be used to model the performance of an arbitrary UV-C network. In the future, the flexible code structure allows more features to be added to the simulation.

15. SUBJECT TERMS
UV communication, UV networking, NLOS communication, interference, Network, Cyber and Computational Sciences

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Fikadu T Dagefu
Unclassified	Unclassified	Unclassified	UU	41	19b. TELEPHONE NUMBER (Include area code) 301-394-0405

Contents

List of Figures

List of Tables

Acknowledgments

Mr Reko wishes to thank the DEVCOM Army Research Laboratory for the summer research opportunity.

1. Introduction

There is an ever increasing urbanization, and it is predicted that two-thirds of the world's population is expected to live in urban areas by 2050. This is causing an unprecedented demand for persistent and secure wireless connectivity in complex environments for civilian and military applications. There has been significant advances in developing new wireless communication technologies that provide unique capabilities leveraging different parts of the electromagnetic (EM) spectrum. In order to address the complex and dynamic communication requirements for various applications, the need to intelligently exploit state-of-the-art and emerging communication technologies is also being explored.[1] One particular optical communication technology that has garnered significant attention recently ultraviolet (UV) communication. Specifically, the UV-C band within the EM spectrum (i.e., wavelengths ranging from 200 to $\sim$ 300 nm) have been shown to have unique propagation characteristics that are favorable.[2–4] First, since the size of the molecules in the atmosphere is comparable to these wavelengths, the solar noise is reduced at the ground level. Second, this scattering characteristics allows for a non-line-of-sight (NLOS) communication link to be created via the reflected signal from the atmosphere when a UV-C source and detector are operating on the ground as depicted in Fig. 1. Finally, the line-of-sight (LOS) propagation loss at UV-C is higher compared to radio frequencies, which makes UV-C a good alternative for short-range LOS/NLOS networking that is harder to detect by an adversary farther away.

While there has been significant research focused on developing point-to-point channel characterization and communication performance modeling,[4–7] there is limited work in investigating the deployment of a UV network in realistic environments with nodes having different LOS and NLOS links. A simulation analysis framework for UV networks deployed in complex environments could provide important insights in terms of steering optimization and performance analysis such as achievable data rate, bit error rate (BER), and interference from neighboring nodes as well as characterization of low probability of detection (LPD) when adversarial nodes are present.

In this report, we develop a new simulation-based analysis framework for a UV wireless communication network deployed in a complex environment (e.g., the one shown in Fig. 1). This involves having multiple ($>$2) UV-C transceivers that are

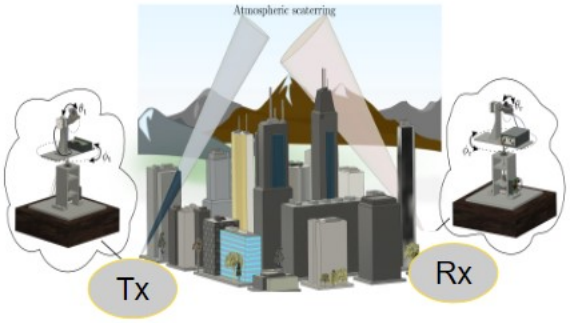

Fig. 1 Depiction of NLOS UV wireless communications in a complex environment [8]

operating in the same communication network. The relevant network parameters include the node locations, transmit powers and beamwidth of the UV sources, and the field of view (FOV) of the UV detectors. Furthermore, the geometry of the complex environment (i.e., the locations of the physical obstructions such as buildings) where the network is deployed is also necessary. The azimuth and elevation steering angles of each transmitter (Tx) and receiver (Rx) pair in the network is then optimized with the goal of maximizing their respective communication links.

For a given network topology deployed in an environment, the simulation code first determines if the transmitter and the intended receiver have an LOS or NLOS link; then it computes the steering angles for optimal communications and provides the resulting performance metrics including the data rate and BER. Furthermore, since multiple transceiver pairs need to communicate simultaneously, the simulation tool computes the interference signals at the intended receiver and takes into account the effect of interference on the communications performance. A user interface (UI) to create different network simulation scenarios and analyze the data is also developed. In the rest of the report, we present background material on UV communication and networking, describe the developed simulation framework and implementation, as well as the UI and show some example simulations. A detailed description of the algorithms and software implementation are also included in Appendixes A and B, respectively.

2. Theoretical Background

2.1 Atmospheric Scattering

Since the sizes of the molecules in the atmosphere are comparable to the wavelength of photons at the UV-C band, this frequency band has favorable characteristics that could be exploited for NLOS wireless communications between nodes that are on or near the ground using scattering from the atmosphere as depicted in Fig. 1. The same scattering characteristics of UV-C with the atmosphere also limits the solar interference for nodes on or near the ground as the UV light at this frequency from the sun is also scattered. Rayleigh and Mie scattering are the two main sources of UV scattering in the atmosphere. Rayleigh scattering consists of light scattering off particles in the atmosphere, while Mie scattering consists of light scattering off the aerosols in the environment.[9,10] The Rayleigh model breaks down with larger wavelengths of light, while the Mie model can be used to approximate the scattering with larger molecules.[10,11] In order to account for the atmospheric scattering in the environment, the Rayleigh and Mie phase functions will be used,[12] expressed as

$$P^{Rayleigh}(\mu) = \frac{3(1+3\gamma+(1-\gamma)\mu^2)}{16\pi(1+2\gamma)} \tag{1}$$

and

$$P^{Mie}(\mu) = \frac{1-g^2}{4\pi}\left[\frac{1}{(1+g^2-2g\mu)^{\frac{3}{2}}} + f\frac{3\mu^2-1}{2(1+g^2)^{\frac{3}{2}}}\right], \tag{2}$$

where $g = 0.72$, $f = 0.5$, and $\gamma = 0.017$. The parameter μ will be defined later and depends on the steering angles of both the Tx/Rx nodes in a communication link. These phase functions determine the distribution of scattering angles of the incident light.[12]

Overall, to account for all the scattering, we combine Eqs. 1 and 2 to create the overall scattering function:

$$P(\mu) = \frac{K_s^R}{K_s^R+K_s^M}P^{Rayleigh}(\mu) + \frac{K_s^M}{K_s^R+K_s^M}P^{Mie}(\mu) \quad . \tag{3}$$

For all these functions, $-1 \leq \mu \leq 1$, the plots showing the scattering functions as a function of μ can be seen in Figs. 2-4 for the case when Mie and Rayleigh scattering are balanced, when Rayleigh scattering dominates, and when Mie scattering is dominant, respectively.

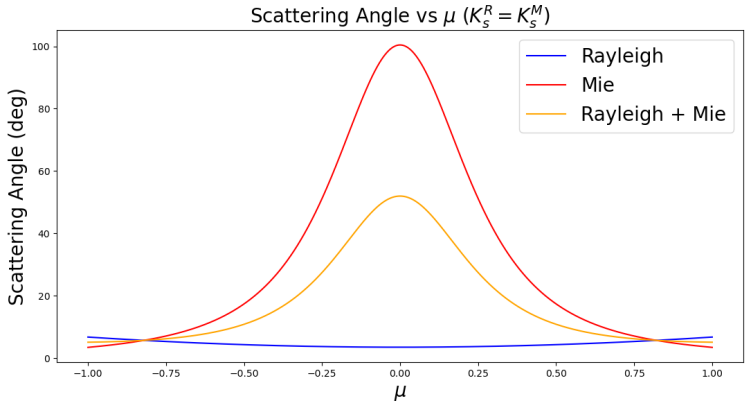

Fig. 2 Scattering functions given equal weights on Rayleigh and Mie scattering

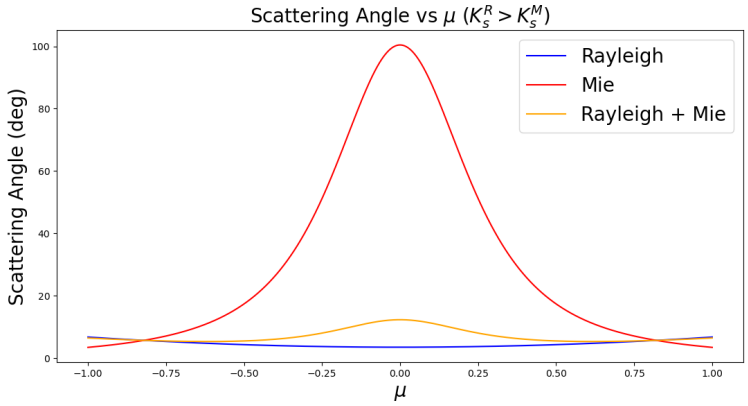

Fig. 3 Scattering functions given $K_s^R = 1$ and $K_s^M = 0.1$

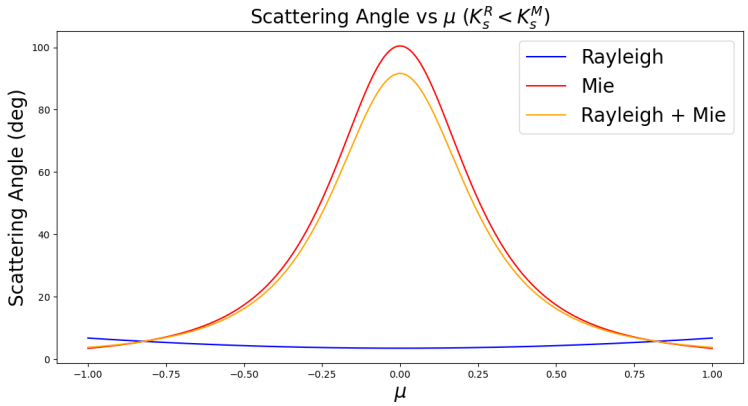

Fig. 4 Scattering functions given $K_s^R = 0.1$ and $K_s^M = 1$

2.2 UV-C Communication Model

The UV-C communication system consists of two main components: 1) a UV-C transmitter, which may be, for example, a UV light-emitting diode (LED) sends photons at the wavelength of interest, and 2) a receiver, which may contain a photon multiplier tube (PMT) combined with a filter to receive the photons along with other photons coming from the different noise sources.The PMT is a device that takes light as an input and outputs varying amounts of current dependent on the number of photons received. A PMT can also be set up with a highly selective band-pass filter that allows only a very narrow band of light to be measured. The LED transmits photons with a uniformly distributed power across its entire beamwidth. The PMT is able to receive photons if they are within its FOV. Overall, the goal is to maximize the overlap between the beamwidth and FOV of the LED and PMT, respectively. The volume of this overlap is directly related to the rate at which photons can be sent. To maximize this overlap, the optimal steering angles (elevation and azimuth) need to be found. This approach is only valid for a single-scatter NLOS model; for multiple scatter models there may be more considerations.

The highest photon rates occur when the communication link is the LOS link. When this occurs, the photon rate is approximately proportional to r^{-2}, where r is the distance between the Tx and Rx nodes. The equation depicting the photon rate can be derived from the received power[4] and is given by

$$\rho_{\text{LOS}} = \frac{\Gamma\lambda}{hc} * \frac{P_t A_r}{4\pi r^2} e^{-K_e r} \ , \tag{4}$$

where ρ_{LOS} is the rate at which photons are received by the PMT, Γ is the efficiency of the PMT, λ is the wavelength from the LED, h is Planck's constant, c is the speed of light, P_t is the transmit power, A_r is the Rx aperture area, r is the distance between the LED and PMT, and K_e is the environmental scattering constant. All other non-steering parameters being equal, the LOS communication channel leads to higher photon rates than the NLOS communication channel.

The NLOS model is more complicated and involves the calculation of the volume of the overlap between the beamwidth and FOV. The exact model[13] uses a specific geometric formulation and relies on the elevation and azimuth angles of both the Tx beamwidth and Rx FOV. In an effort to derive the exact intersection volume, a triple integral using polar coordinates is required.[13] The software implementation of this

exact volume calculation works efficiently if the steering angles are known. However, in an effort to find the optimal angles through a trust-constrained optimization technique,[14] the integral calculation becomes too computationally expensive.

3. Proposed Approach for UV-C Network Simulation

3.1 Design of Simulation Framework

The diagram shown in Fig. 5 depicts the components of the proposed UV-C network simulation tool. The required inputs are shown on the left side of the diagram in Fig. 5. This includes the parameters describing the Tx and Rx nodes such as the locations, transmit power, beamwidth, and FOV. A CAD file containing the scene geometry is also a required input.

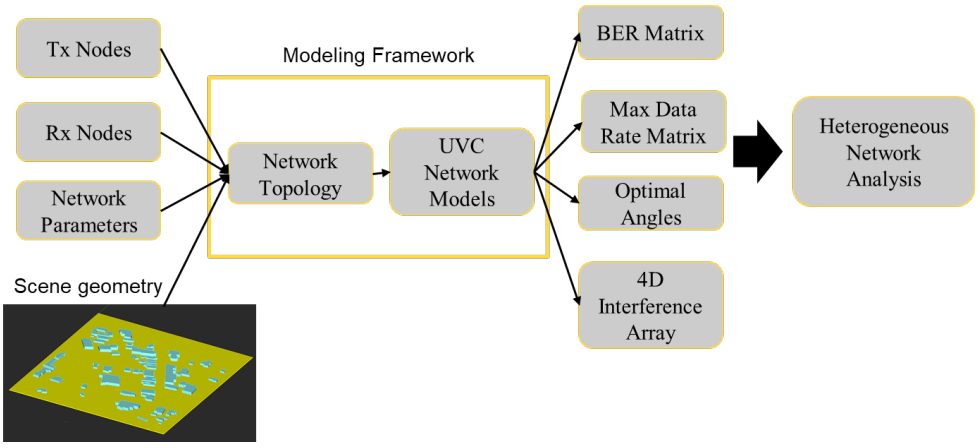

Fig. 5 Components of the proposed UV-C network simulation framework

Based on the above inputs, the network topology is created and overlayed on the scene geometry. Then, the LOS and NLOS links are determined for all pairs of Tx and Rx nodes. After that, the optimization procedure described in the next section is used to determine the steering angles of the Tx and Rx nodes. The last step in the computation of performance metrics includes BER, data rate, as well as interference.

3.2 Steering Optimization and Communication Performance

As mentioned in the previous section, the computation of the photon rates for NLOS scenario is computationally expensive.[13] For that reason, we use an approximation of the exact model[5] to reduce the number of computations. The geometric formu-

lation of the approximated model[5] can be seen in Fig. 6 and the model uses the Legendre-Gaussian quadrature approximation.[3] The implementation of the model algorithm from Zuo et al.[14] can be seen in Appendix A.

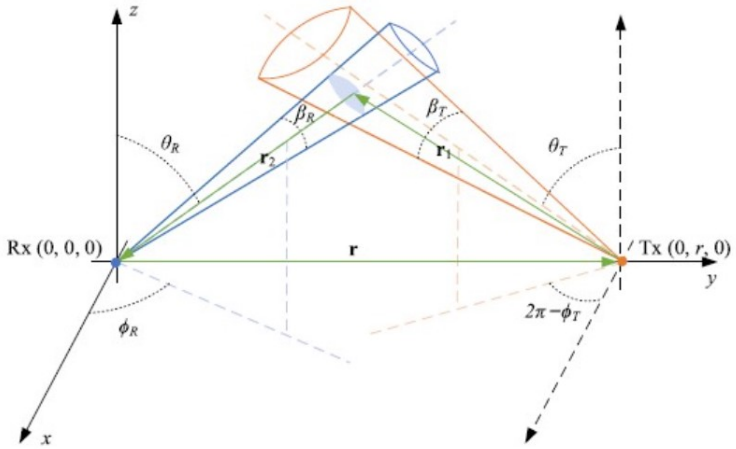

Fig. 6 NLOS geometric formulation[14]

For the optimization of the UV-C NLOS model, the communication link must fit into the geometry of Fig. 6. This means that the receiving node is at the origin and the transmitting node is at (0, d, 0), where d is the distance from Tx to Rx. When the two nodes are aligned in azimuth, then $\phi_T = -90°$ and $\phi_R = 90°$ for the Tx and Rx azimuth angles, respectively. The optimization is also constrained by physical obstructions in the network. These come in the form of buildings, walls, or any other physical obstacle that interrupts the channel. More on how the elevation angles are constrained is detailed in Section 4.

Once we implemented a steering optimization approach and a way to efficiently compute the photon rates, the next step was to develop an approach for communication performance analysis. During the course of this study, we considered a few different modulation schemes. However, due to time limitation, we only implemented on-off-keying (OOK). OOK involves setting a threshold value for received photons and if the PMT detects that more photons have been detected compared to the threshold, it is considered as '1'. Otherwise, a '0' is transmitted.

3.3 Interference Computation

Interference between multiple UV-C communication links acting simultaneously is a less often studied concept. The theoretical model detailed in Section 2.2 enables

a simulation-based analysis. The key requirement is finding the azimuth angles of two nodes when they are configured for communication with other nodes. The algorithm is described as the interference algorithm in Appendix A. An additional factor in the interference problem is the presence of physical obstructions blocking signals. This is accounted for in the software implementation. Figure 7 shows how the interference changes when either an interfering Tx or an intended Rx is rotated. As the variable ρ is rotated in the intended direction, for either the Tx or Rx, the interfering photon rate will change as shown on the right of Fig. 7. The blue plot corresponds to Tx2 being rotated toward the intended receiver, Rx1, that has a steering direction for an optimized communication link with Tx1. The red plot shows what happens when Rx1 is rotated to point toward Tx2 when Tx2 has a steering direction optimized for communication with Rx2. The orange rectangle is a wall that blocks LOS communication links between the pairs Tx1/Rx1 and Tx2/Rx2.

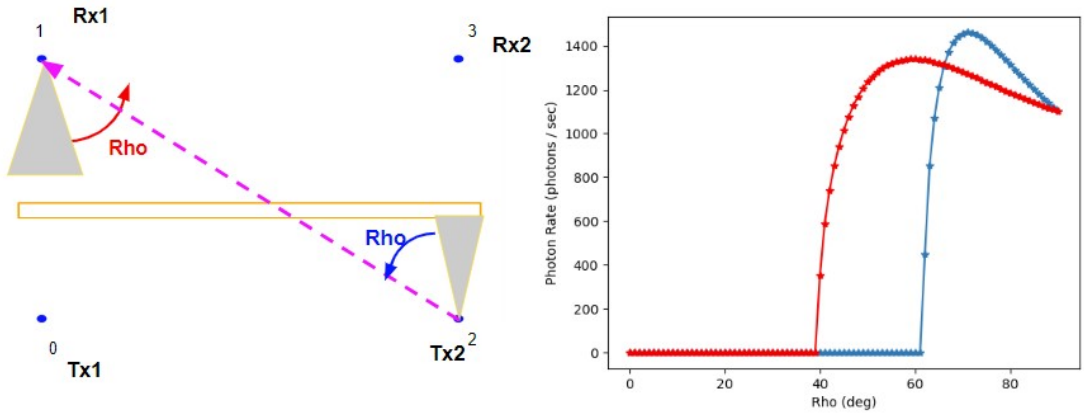

Fig. 7 UV-C Interference with rotating Tx beamwidth and Rx FOV

4. Software Implementation

4.1 Network Topology

Using the software structure from the previous section, this section describes the way to implement the model given the imported file `UVC_Experiments.py`. The first step is to load in UV-C transceiver nodes. This can be done by declaring multiple instances in code and then putting those instances in a list. Additionally, there is a function `read_node_csv` that takes a .txt file as input and creates instances of `tx_node` and `rx_node`. The .txt file needs to be created so it follows

the format in Fig. 8, where each line or row contains the following comma-separated parameter entries in order.

1. Tx Power
2. Tx beamwidth
3. Tx elevation angle
4. Tx azimuth angle
5. X coordinate
6. Y coordinate
7. Tx wavelength
8. Data rate
9. Rx FOV
10. Rx elevation angle
11. Rx azimuth angle
12. Rx efficiency
13. Aperture area

```
0.001, 17, 30, 0, 0, 0, 275e-9, 10e4, 37, 30, 0, 1, 10e-4
0.001, 17, 30, 0, 5, 0, 275e-9, 10e4, 37, 30, 0, 1, 10e-4
0.001, 17, 30, 0, 10, 0, 275e-9, 10e4, 37, 30, 0, 1, 10e-4
0.001, 17, 30, 0, 15, 0, 275e-9, 10e4, 37, 30, 0, 1, 10e-4
0.001, 17, 30, 0, 20, 0, 275e-9, 10e4, 37, 30, 0, 1, 10e-4
0.001, 17, 30, 0, 25, 0, 275e-9, 10e4, 37, 30, 0, 1, 10e-4
```

Fig. 8 Example of .txt file with transceiver characteristics

Additionally, physical obstructions need to be loaded into the network. This is done by loading two .txt files with physical obstruction vertices and heights. The vertices and heights .txt files are loaded with `read_obs_txt` and `read_obs_h_csv` functions, respectively. The .txt file formats can be seen in Fig. 9, and it is critical for the vertices that there are no spaces inside the ordered pairs and no commas between ordered pairs.

Overall, the formulation of the network topology is done after loading in the transceiver, obstructions, setting the noise, and setting the constraints of each node (usually set so the transceiver can access all possible steering angles). The lower bound of the constraint is always changed when optimization algorithms are run on the network,

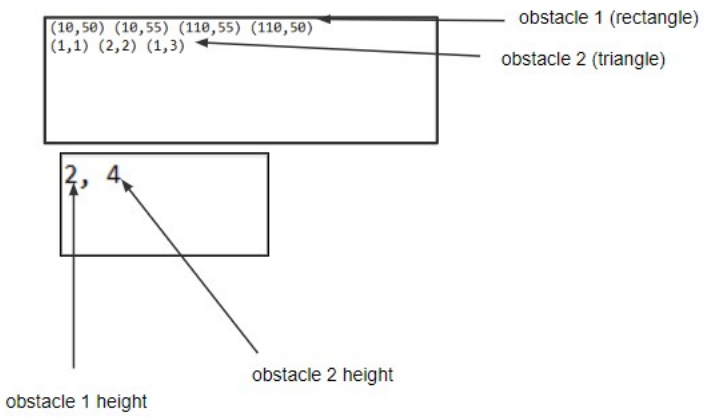

Fig. 9 Example of .txt files with obstruction vertices (top) and heights (bottom)

```
from UVC_Experiments import *

tx_arr, rx_arr = read_node_csv('nodes16.txt')  # Load nodes through nodes16.txt file
obs_array = read_obs_txt('load_obs.txt')  # Load obstruction vertices from load_obs.txt
h1 = read_obs_h_csv('load_h.txt')  # Load obstruction heights from load_h.txt
constraints = []
noise = 1000  # Set ambient noise in environment
for i in range(len(tx_arr)):
    tmpLb = [np.pi / 6, -np.pi, 0, np.pi / 6]  # Irrelevant, changed with optimization
    tmpUb = [np.pi, np.pi, np.pi, np.pi]  # Allows for all angles to be met
    A = [[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]]  # Identity matrix
    constraints.append((tmpLb, tmpUb, A))  # Add the constraints for all nodes in tx_arr
network1 = txrx_network(tx_arr, rx_arr, constraints, noise)  # Network declaration
network1.load_obstacles(obs_array, h1)  # Network topology has physical obstructions loaded in
```

Fig. 10 `txrx_network` class declaration code

so they are ignored whenever those algorithms are used. The code in Fig. 10 shows how to initially declare a `txrx_network` instance.

4.2 Run Models

After loading a network in, calling the functions detailed in Appendix B.2 is the next step in creating data that can be stored in a .mat file and used for further study. The first function to call is `opt_General_sim_geom` as it will set the `photon_mat` and `angle_mat` matrices in the instance of `txrx_network` to the optimal values. This means that the steering angles are now set to have the maximum received power for every Tx/Rx connection in the network topology. After this, the interference matrix is created by calling `create_interference_matrix`; this 4D interference matrix is detailed earlier. After this, the functions detailed in Appendix B.2 that create a signal-to-noise

ratio (SNR) matrix, find the signal-to-interference-plus-noise ratio (SINR) of a specific scenario, and find the maximum data rate dependent on SNR and BER thresholds can be used. The code to do this can be seen in Fig. 11.

```
network1 = txrx_network(tx_arr, rx_arr, constraints, noise)  # Network declaration
network1.load_obstacles(obs_array, h1)  # Network topology has physical obstructions loaded in
Ka = 1e-4  # Atmospheric absorption constant (arbitrary value used here)
Ks_r = 1e-4  # Atmospheric Rayleigh scattering constant (arbitrary value used here)
Ks_m = 1e-4  # Atmospheric Mie scattering constant (arbitrary value used here)
plot_network_topology(network1)  # Uses matplotlib.pyplot to plot obstructions and node locations
photon_rate, te1, re1, tp1, rp1 = opt_General_sim_geom(network1, Ka, Ks_r, Ks_m)  # Find optimal angles/photon rates
create_interference_matrix(network1, Ka, Ks_r, Ks_m)  # Create 4D interference array
network1.calc_SNR()  # Calculate SNR and BER for every pair in network
network1.calc_SINR([(0, 1), (2, 4), (3, 5)])  # Find SINR and BER from 0-->1 when 2-->4 and 3-->5 are also active
network1.SNR_threshold = 1  # min SNR in dB
network1.BER_MAX = 1e-5  # max BER
network1.find_max_rate()  # Finds max rate dependent on SNR_threshold and BER_MAX
```

Fig. 11 Implementation of UV-C models in code

4.3 Other Functionality

Other than running the models in `UVC_Experiments.py`, the data can be manually altered to reflect a change in transceiver characteristics, steering angle, or elevation constraints. Examples of this can be seen in the file `UVC_Experiments_Testing.py`.

4.4 Scene Geometry

The use of computer-aided-design (CAD) allows for a more complicated network topology to be loaded in more easily. In order to account for this need, another Python file is available called `OCC_test.py`. This file takes a .stp file and finds all the vertices and heights of the buildings in the topology. This allows for a .txt file to be input into the current model as shown in Section 4.1. This code is only able to accurately map convex-faced obstructions into the appropriate file type. This is due to the vertices not being ordered when the CAD file is read, leading to no indication of the vertex order. The method to order the points requires finding the center and ordering the points based on distance from the center. For non-convex-faced obstructions, the code is not guaranteed to order the vertices correctly. In order to do that, an algorithm designed to solve the traveling salesman problem must be used. The output of this script can be seen in Fig. 12.

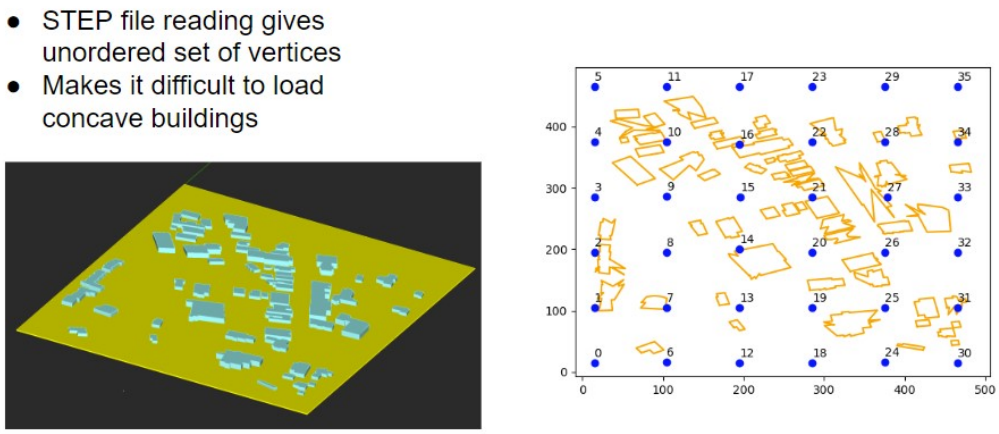

Fig. 12 Importing the scene geometry

5. User Interface

To help users navigate the code, a simple UI has been created. The UI main screen shown in Fig. 13 is designed in an order that mirrors Section 4.1 and 4.2. The Python file containing the UI is `UVC_UI2.py` and relies on the PyQt5 library to create the interface.

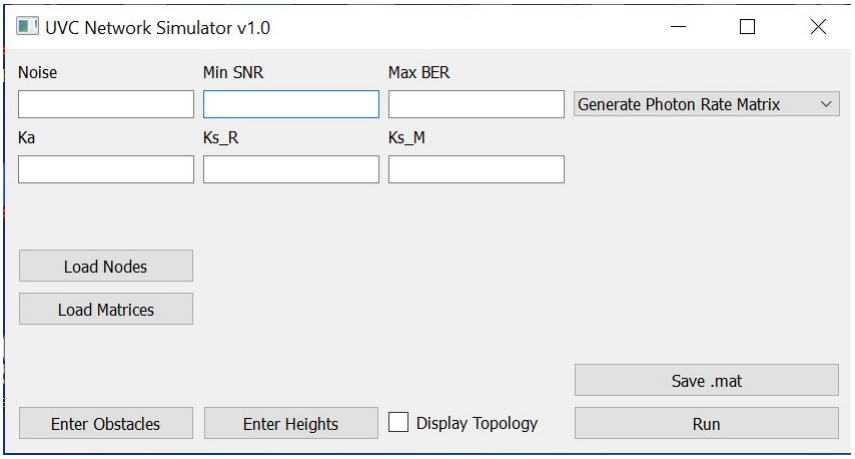

Fig. 13 UI main screen

The first step to use the UI is to fill in all the blank boxes with information about noise, min SNR, max BER, K_a, K_s^R, and K_s^M as they have been used previously. Then, clicking the button labeled "Load Nodes" will pull up the file system (only tested on Windows) and the appropriate .txt file can be selected for the desired transceiver nodes.

After loading the node information, the options are to run LOS models or load obstacles with obstacle heights. To do the latter, the buttons labeled “Enter Obstacles“ and “Enter Heights” need to be selected. Then the appropriate .txt files for the vertices and heights can be loaded. To ensure that the topology is correct, the “Display Topology” option should be selected, which will generate a 2D plot of the network after the “Run” button is selected.

Once the network is loaded in, experiments can be conducted. This involves selecting the drop-down menu on the top right and selecting one of the options shown in Fig. 14. The options for models can be seen in Table 1.

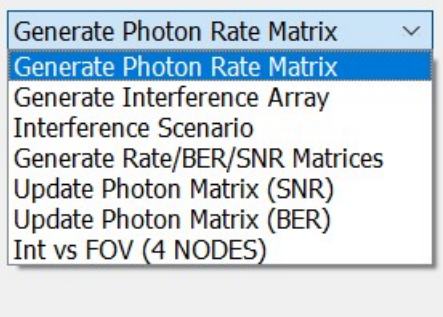

Fig. 14 UI drop-down menu to select experiment

Table 1 UI model options

Menu Name	Description
Generate Photon Rate Matrix	Finds optimal photon rate for every possible Tx/Rx pair in network
Generate Interference Array	Creates 4D interference array
Interference Scenario	Finds SINR in a user input scenario
Generate Rate/BER/SNR Matrices	Calculates matrices corresponding to max data rate, BER and SNR
Update Photon Matrix (SNR/BER)	Removes communication if not meeting SNR/BER requirements
Int vs FOV	Finds how FOV affects interference in 4-node case only

For the Interference Scenario option, an initial screen asks how many communication links are active at the same time. After this is answered, another screen (Fig. 15) comes up, dictating which nodes that are to be investigated. The first row is the pair for which the SINR will be calculated; the other rows are designated for interfering links. Additionally, the value to be stored is determined by the metric being tracked, which can be changed by the drop-down menu at the bottom.

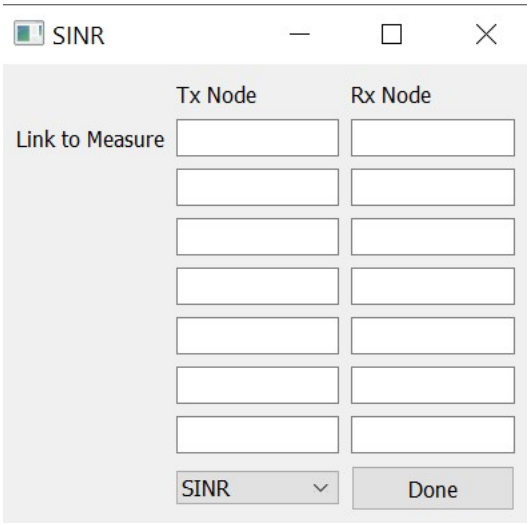

Fig. 15 UI screen to select link of interest

For the Update Photon Matrix options, the photon rate matrix that is outputted can be chosen such that only nonzero entries reflect when the communication links satisfy either the SNR or BER requirement.

For the Int vs FOV option, it can be seen how increasing the FOV of node 3 in the network impacts the signal from $0 \rightarrow 3$ while node 1 communicates to node 2. This specific order enables finding the the signal-to-interference ratio (S/I) of a 4-node network and how the FOV can impact that metric. The order of the nodes described is the only way that the nodes are configured for this option.

After running the models (the recommended action is to run the first two options in the drop-down menu and then proceed to the others), a .mat file containing information detailed in Table 2 can be saved to the computer. This allows further use in MATLAB or by using the numpy Python library. After this is done the first time, the photon rate matrix and 4D interference array can by reloaded and the other experiments run by loading in the same .mat file after clicking the "Load Matrices" button.

6. Example Simulation Results

After the software implementation, various simulation examples were performed. We describe some of the results in this section. The single link results were used to validate the models used in Cao et al.,[4] and the results match those in other works.[5,8]

Table 2 Input and output file names

Mat File Variable Name	Description
`ang_mat`	Steering angle matrix
`photon_mat`	Photon rate matrix
`int_mat`	4D interference array
`interference_vector`	List of all SINR/BER scenario results performed
`Min_SNR`	Minimum SNR used in UI
`Max_BER`	Maximum BER used in UI
`SNR_mat`	SNR values for every pair
`SNR_mat_dB`	SNR values for every pair in dB
`BER_mat`	BER values for every pair
`Rate_mat`	Maximum data rate values for every pair given SNR requirement
`Rate_mat_BER`	Maximum data rate values for every pair given BER requirement
`INTvFOV`	Interference vs FOV vector in 4-node case
`SIGvFOV`	Signal vs FOV vector in 4-node case

For the networking results, the topology will involve many specific scenarios, but the same code can be used for any network topology. The biggest difference between running the models with varying topology is the run time. If there are a lot of NLOS links in the network, the code will take longer. It is recommended to use these models for networks with a maximum size of 36 nodes.

6.1 FOV vs. Interference

As the FOV increases, more signal power is able to be received by the Rx node. However, in a scenario where two communication links are acting simultaneously, an increase in FOV size allows more interfering photons to be received as well. In order to find the S/I, we look at the 4-node case where the communication links cross (Fig. 16). It can be seen how the S/I ratio decreases as the FOV is increased. This ultimately leads to the conclusion that the best FOV option is the minimum FOV to ensure stable communication.

6.2 Maximum Data Rate in Realistic Network

As shown in Fig. 17, the maximum data rate for a specific scenario changes depending on the network topology between nodes. All the buildings are the same height (5 m) in this scenario. The LOS nodes have a higher maximum data rate than the NLOS nodes and have a near quadratic relationship with distance from node 0.

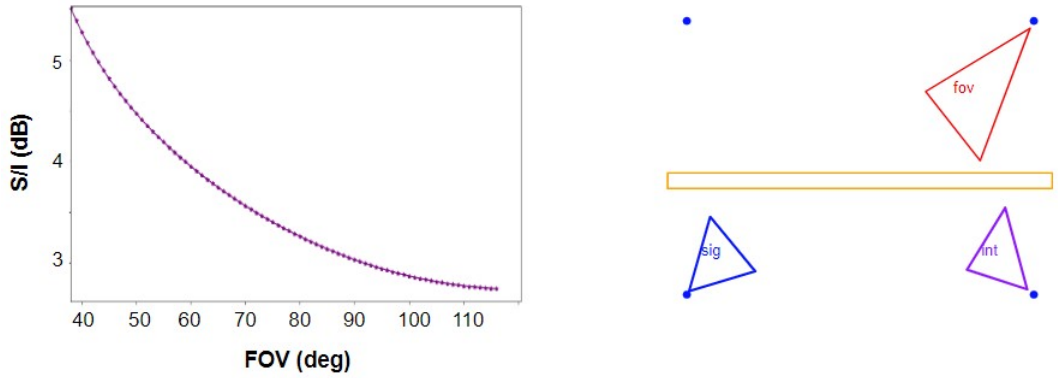

Fig. 16 S/I for specific scenario where FOV increases

The NLOS connections may not be usable if they are separated by too much distance. Overall, this allows the determination of which links are capable of handling different communication requirements (e.g., audio, video) in practical scenarios. For communication across the network, a half-duplex, multi-hop strategy may be needed.

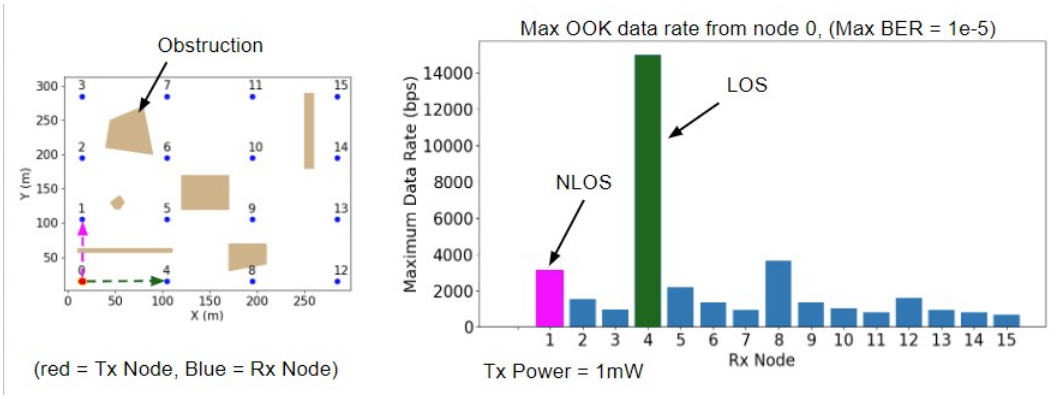

Fig. 17 UV-C data rate in realistic network topology

6.3 Interference Analysis

In order to analyze simultaneous operation of UV-C communication links, a desired communication link and interfering links need to be specified. This allows for the calculation of the SINR and BER for the desired link. In the same topology, certain communication links do not have a significant impact on the BER from node 6 to node 9 (Fig. 18). However, certain links do cause a near order of magnitude BER increase. Overall, for an arbitrary network configuration, it is possible to see

how the performance degrades as more links are used simultaneously. This is done with each communication link being optimized for just that link, there is no joint optimization in this scenario.

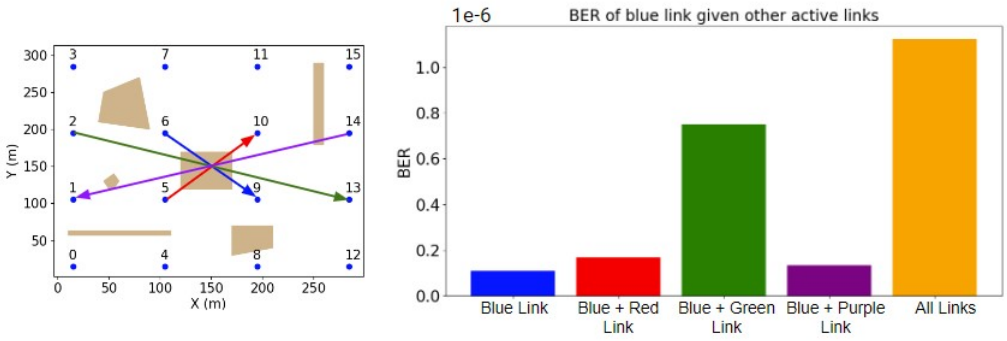

Fig. 18 BER increase in the presence of interference

7. Conclusions and Future Work

Overall, the UV-C network simulation model allows for the analysis of an arbitrary network of UV-C transceivers. The simulation model includes an approximation of the theoretical UV-C communication model and yields the pairwise photon rates, SNR, BER, and maximum data rates. The analysis results could provide 1) evaluation of the capabilities the UV-C networking modality, 2) future application use (e.g., as part of a heterogeneous networking paradigm), and 3) an exploration of the possible interference effects. The developed model allows expert users to configure their network for the desired application and characterize the communication performance degradation due to interference effects as well as to inform the development of intereference mitigation techniques.

In the future, a number of efforts could be pursued to improve the developed analysis tool. First, a feature can be developed to enable the joint optimization of multiple communication links within the framework. Rather than just optimizing the UV-C NLOS links, the interference results can be combined and the S/I optimized to achieve a more optimal communication performance. Second, the transceiver nodes are currently in a 2D plane. Extending this to 3D by incorporating a new communication model, such as the one discussed in Cao et al.[4] . The third idea is to integrate a variety of modulation approaches in the simulation framework, in addition to OOK, so that a variety of modulation approaches can be analyzed. The fourth idea

is to remove the assumption in the current version of the simulation tool in terms of the beamwidth and FOV being fully clear of any physical obstructions. There is a possibility of a hybrid LOS/NLOS link, where only part of the beamwidth/FOV is blocked by the obstruction. It would be useful to add the simulation capability for such cases. Finally, more networking features can be added to understand data flows across a network.

8. References

1. Kong J, Dagefu FT, Choi J, Spasojevic P, Koumpouzi C. Covert communications in low-vhf/microwave heterogeneous networks. In: 2022 IEEE Wireless Communications and Networking Conference (WCNC); 2022 Apr 10–13; Austin, TX, USA. p. 1970–1975.

2. Luettgen MR, Shapiro JH, Reilly DM. Non-line-of-sight single-scatter propagation model. JOSA A. 1991;8(12):1964–1972.

3. Arslan CH, Dagefu FT, Moore TJ, Weisman MJ, Drost RJ. Measurement system for ultraviolet channel modeling and communications. Optics Express. 2023;31(15):23714–23728.

4. Cao T, Song J, Pan C. Simplified closed-form single-scatter path loss model of non-line-of-sight ultraviolet communications in noncoplanar geometry. IEEE Journal of Quantum Electronics. 2021;57:1–9.

5. Arslan CH, Dagefu FT, Weisman MJ, Drost RJ. Channel model validation for and extensions of an ultraviolet networking optimization framework. In: 2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall); 2019 Sep 22–25; Honolulu, HI, USA. p. 1–7.

6. Ding H, Chen G, Majumdar AK, Sadler BM, Xu Z. Modeling of non-line-of-sight ultraviolet scattering channels for communication. IEEE Journal on Selected Areas in Communications. 2009;27(9):1535–1544.

7. Xu Z, Sadler BM. Ultraviolet communications: potential and state-of-the-art. IEEE Communications Magazine. 2008;46(5):67–73.

8. Arslan CH, Dagefu FT, Moore TJ, Weisman MJ, Drost RJ. Optimization of ultraviolet communication links based on finite difference stochastic approximation. Optics Express. 2022;30(20):36283–36296.

9. Twersky V. Rayleigh scattering. Applied Optics. 1964;3(10):1150–1162.

10. Hahn DW. Light scattering theory. Department of Mechanical and Aerospace Engineering, University of Florida; 2009.

11. Zhang Q, Zhang X, Wang L, Shi G, Fu Q, Liu T. Performance modeling of ultraviolet atmospheric scattering of different light sources based on monte carlo method. Applied Sciences. 2020;10(10):3564.

12. Zachor AS. Aureole radiance field about a source in a scattering–absorbing medium. Applied Optics. 1978;17(12):1911–1922.

13. Xu Z. Approximate performance analysis of wireless ultraviolet links. In: 2007 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'07); 2007 Apr 15–20; Honolulu, HI, USA. p. III–577.

14. Zuo Y, Xiao H, Wu J, Li Y, Lin J. A single-scatter path loss model for non-line-of-sight ultraviolet channels. Optics Express. 2012;20(9):10359–10369.

Appendix A. Algorithms

A.1 UV-C NLOS Model Algorithm

The ultraviolet-C (UV-C) non-line-of-sight (NLOS) model algorithm outputs the received power P_r from which the received photon rate can be calculated as $\rho_{\text{NLOS}} = P_r \times \frac{\Gamma\lambda}{hc}$. The required inputs are the transmitter (Tx) and receiver (Rx) elevation and azimuth angles θ_T, ϕ_T, θ_R, and ϕ_R; the transmit power P_t; the distance r from the Tx to the Rx; the transmitter beamwidth β_T and receiver field of view (FOV) β_R; the receiver aperture area A_r; the atmospheric absorption constant K_a; the Rayleigh and Mie scattering constants K_s^R and K_s^M; and the scattering function $P(\mu)$ in Eq. 3 of the main report.

Algorithm 1 UV-C NLOS Model Algorithm

Inputs: θ_T, ϕ_R, θ_R, ϕ_R, P_t, r, A_r, β_T, β_R, K_a, K_s^R, K_s^M, and $P(\mu)$
Outputs: P_r
$\tilde{\theta}_T \leftarrow \frac{\pi}{2} - \theta_T$
$\tilde{\theta}_R \leftarrow \frac{\pi}{2} - \theta_R$
$a \leftarrow \left[\sin\tilde{\theta}_T \sin\tilde{\theta}_R \cos\left(\phi_T - \phi_R\right) + \cos\tilde{\theta}_T \cos\tilde{\theta}_R\right]^2 - \cos^2\frac{\beta_R}{2}$
$b \leftarrow 2r\sin\phi_T\left[\sin^2\tilde{\theta}_R \sin\phi_R \cos(\phi_T - \phi_R) - \sin\phi_T \cos^2\frac{\beta_R}{2}\right]$
$\quad + r\cos\tilde{\theta}_T \sin 2\tilde{\theta}_R \sin\phi_R$
$c \leftarrow r^2\left(\sin^2\tilde{\theta}_R \sin^2\phi_R - \cos^2\frac{\beta_R}{2}\right)$
$\vec{\mu_R} \leftarrow [\sin\tilde{\theta}_R\cos\phi_R, \sin\tilde{\theta}_R\sin\phi_R, \cos\tilde{\theta}_R]^{\mathbf{T}}$
$\alpha \leftarrow \frac{\sqrt{b^2-4ac}}{2a}$
$\eta \leftarrow \frac{\sqrt{-b}}{a}$
$\epsilon \leftarrow 0$
if $Im\{\alpha\} = 0$ **then**
  for $0 \le k \le n$ **do**
    $\nu \leftarrow \eta + \alpha t_k$
    $\vec{r_1} \leftarrow [\nu\sin\tilde{\theta}_T\cos\phi_T, \nu\sin\tilde{\theta}_T\sin\phi_T, \nu\cos\tilde{\theta}_T]^{\mathbf{T}}$
    $\vec{r_2} \leftarrow [0, -r, 0]^{\mathbf{T}} - \vec{r_1}$
    $\mu \leftarrow \frac{\vec{r_1}\cdot\vec{r_2}}{||\vec{r_1}||\cdot||\vec{r_2}||}$
    $\xi \leftarrow \frac{-\vec{r_2}\cdot\vec{\mu_R}}{||\vec{r_2}||}$
    $\epsilon \leftarrow \epsilon + \omega_k \frac{A_r(K_s^R+K_s^M)\xi P(\mu)\exp\{-(K_a+K_s^R+K_s^M)(||\vec{r_1}||+||\vec{r_2}||)\}}{2||\vec{r_2}||^2\left(1-\cos\frac{\beta_T}{2}\right)} \tan^2\frac{\beta_T}{2}$
  end for
  $P_r \leftarrow P_t \times \alpha \times \epsilon$
else
  $P_r \leftarrow 0$
end if

A.2 UV-C Interference Algorithm

The UV-C Interference algorithm outputs the rate of photons $\rho_{\text{Interference}}$ at the desired Rx from the interfering Tx. The required inputs are the intefering Tx and affected Rx elevation and azimuth angles θ_T, ϕ_T, θ_R, and ϕ_R; the transmit power P_t of the interferer; the distance r from the interfering Tx to the Rx; the interfering transmitter beamwidth β_T and receiver FOV β_R; the receiver aperture area A_r; the atmospheric absorption constant K_a; the Rayleigh and Mie scattering constants K_s^R and K_s^M; the scattering function $P(\mu)$ in Eq. 3 of the main report; and the azimuth angle vector between the interfering Tx and Rx nodes ψ_T and ψ_R.

Algorithm 2 UV-C Interference Algorithm

1: Inputs: θ_T, ϕ_T, θ_R, ϕ_R, P_t, r, A_r, β_T, β_R, K_a, K_s^R, K_s^M, $P(\mu)$, ψ_T, and ψ_R
2: Output: The rate of photons at the desired receiver from the interfering transmitter
3: $\phi_1 \leftarrow \frac{-\pi}{2} + \phi_T - \psi_T$
4: $\phi_2 \leftarrow \frac{\pi}{2} + \phi_R - \psi_R$
5: $P_r \leftarrow$ UVC NLOS $(\theta_T, \phi_1, \theta_R, \phi_2, P_t, r, A_r, \beta_T, \beta_R, K_a, K_s^R, K_s^M, P(\mu))$
6: $\rho_{\text{Interference}} \leftarrow P_r \times \frac{\Gamma\lambda}{hc}$

Appendix B. Software Description

To implement the approaches described in Section 3 of the main report, three Python files were used to compose the back end calculations and a fourth Python script implemented the user interface (UI). The back end is composed of the files: `UVC_Functions.py`, `UVC_Simulation_Classes.py`, and `UVC_Experiments.py`. All of these will be detailed in the following subsections in order of implementation, which means that every subsequent file imports the previous file.

B.1 `UVC_Functions`

This file contains the algorithms depicted in Appendix A, as well as helper functions for those algorithms. The different functions are detailed in the following subsections.

B.1.1 `NLOS_closed_form_txrx`

Software implementation of UV-C NLOS Model Algorithm from Appendix A of this report. It outputs $-\ln(P_r)$, as this helps speed up the optimization. To undo this for actual power values, perform e^{-out} when out is the output of the function. To avoid computer precision error, $e^{-1000} = 0$, making it synonymous with the algorithm.

B.1.2 `check_cone_intersection`

Uses cone intersection algorithm[1] to see if two cones intersect. Integrated in `NLOS_closed_form_txrx`, so not used elsewhere. Could be useful for future development.

B.1.3 `find_initial_az_angles`

Finds the azimuth angle between two points in space. This assumes a universal coordinate system. This function outputs the values of ψ_T, ψ_R from the interference algorithm in Appendix A.

B.1.4 `opt_NLOS_closed_form_geom`

Takes in the same arguments as `NLOS_closed_form_txrx` as well as a linear constraint from the scipy.optimize library. The linear constraint is applied to all the steering angles of the transmitter (Tx) and receiver (Rx) node and constrains the optimization of `NLOS_closed_form_txrx`. The optimization uses the scipy.optimize.minimize function, thus why we define the output of `NLOS_closed_form_txrx`

[1] Twersky V. Rayleigh scattering. Applied Optics. 1964;3(10):1150–1162.

Table B-1 `tx_node` Class variables

Variable Name	Variable Description
`tx_power`	Tx power
`bw`	Tx beamwidth
`elev_angle`	Tx elevation angle (θ_T)
`azimuth_angle`	Tx azimuth angle (ϕ_T)
`X`	x-coordinate of transmitter
`Y`	y-coordinate of transmitter
`wavelength`	Transmitted wavelength (λ)
`R`	Tx data rate

Table B-2 `rx_node` Class variables

Variable Name	Variable Description
`fov`	Rx FOV
`elev_angle`	Rx elevation angle (θ_R)
`azimuth_angle`	Rx azimuth angle (ϕ_R)
`X`	x-coordinate of receiver
`Y`	y-coordinate of receiver
`efficiency`	PMT efficiency
`A_r`	Aperture area

as we did. The minimization function also takes an initial guess as an argument, so this defaults to `min_tx_elev` and `min_rx_elev` as the initial elevation guess for the Tx and Rx node, respectively. For the initial azimuth guesses, -90° and 90° should be used for the Tx and Rx nodes, respectively.

B.2 `UVC_Simulation_Classes`

This file contains the primatry classes that are used in the model. How to initialize these classes is detailed in Section 4 of the main report. The classes and relevant class functions are detailed in the following subsections.

B.2.1 `tx_node`

This class stores the data for a transmitting node. The class has no functions, but stores the internal variables listed in Table B-1.

B.2.2 `rx_node`

This class stores the data for a receiving node. The class has no functions, but stores the internal variables listed in Table B-2.

Table B-3 `txrx_single_link` Class variables

Variable Name	Variable Description
`Tx`	Instance of tx_node class
`Rx`	Instance of rx_node class
`Rx_photons`	Received photon rate in link (default=0)
`Rx_power`	Received power in link (default=0)
`constraints`	scipy `LinearConstraint` for steering angles (default=no constraint)

B.2.3 `txrx_single_link`

This class uses the `tx_node` and `tx_node` classes to perform calculations for a single communication link. It allows for the implementation of the models from `UVC_Functions` to find communication results. The initialization takes two parameters, one Tx node and one Rx node, which are instances of the `tx_node` and `rx_node` classes, respectively. The internal variables and functions are listed and described in Table B-3.

The `txrx_single_link` class includes the following functions:

`LOS_photon_count_Xu`: Uses model from Xu[2] to find the LOS photon rate and store in the internal variables from Table B-3. Same equation as seen in Section 2 of the main report.

`NLOS_photon_count_Xu`: Uses the non-line-of-sight (NLOS) model from Xu[2] and computes the received power given arbitrary steering angles. These models from Xu[2] assume that the nodes are pointing at each other in azimuth and only calculate the power based on the elevation angles. This function stores the results in the internal variables.

`optimize_NLOS_angles_Xu`: Optimizes the NLOS equation from Xu[2] and returns the optimal elevation angles given a constraint. This also stores the photon rate and power in the appropriate internal variables.

`opt_NLOS_General_geom`: Optimizes the NLOS equation from Zuo et al.[3] and returns the optimal elevation/azimuth angles given a constraint. This also stores the

[2]Xu Z. Approximate performance analysis of wireless ultraviolet links. In: 2007 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP' 07); 2007 Apr 15–20; Honolulu, HI, USA. p. III–577.

[3]Zuo Y, Xiao H, Wu J, Li Y, Lin J. A single-scatter path loss model for nonline- of-sight ultraviolet channels. Optics Express. 2012;20(9):10359–10369.

Table B-4 `txrx_network` Class variables

Variable Name	Variable Description
`tx_nodes`	List of instances of `tx_node` class
`rx_nodes`	List of instances of `rx_node` class
`const`	List of `LinearConstraint` instances for each node
`noise`	Ambient noise in the network
`obstacles`	List of vertices of physical obstructions in the network (default=[])
`obstacle_heights`	List of obstacles heights in network (default=[])
`isLOS`	List of boolean values indicating if a link is LOS or not
`links`	List of `txrx_single_link` instances (default=[])
`all_links`	List of all possible `txrx_single_link` instances (default=[])
`photon_counts`	List of photon rates for all links in `links` (default=[])
`links_created`	Set to True if communication links have been set (default=False)
`angle_mat`	numpy array containing all steering angles in network
`photon_mat`	numpy array containing photon rates for every Tx/Rx pair in network
`interference_mat`	A 4D numpy array containing interference information
`SNR_threshold`	Sets a minimum SNR value (default=$-\infty$)
`BER_MAX`	Sets a max BER value (defualt=0.5)
`SNR_mat_dB`	numpy array of SNR values in dB for all links in network
`BER_mat`	Matrix of BER values for all possible links
`rate_mat`	Matrix of maximum data rate values dependent on `SNR_threshold`
`rate_mat_ber`	Matrix of maximum data rate values dependent on `BER_MAX`

photon rate and power in the appropriate internal variables.

`create_angle_constraint`: Creates a new `scipy.optimize.LinearConstraint` with the inputs of an upper bound, lower bound, and equality matrix. These are then input into a new `LinearConstraint` instance that overwrites the default.

`add_noise`: Adds an integer value of ambient noise to the communication link.

B.2.4 `txrx_network`

This class adds a set of Tx and Rx nodes and creates a network with an arbitrary topology. Obstacles can be loaded in to change the constrained optimization algorithm and data is stored in the network. The initial inputs to this class are a list of Tx nodes, list of Rx nodes, list of steering constraints, and ambient noise value. This is the biggest class and keeps track of all the models data. The important internal variables and class functions are listed and described in Table B-4. There are other internal variables, but their use is clear in the code.

The `txrx_network` class includes the following class functions:

`load_photon_angle_matrices`: Overwrites the values stored in `txrx_network.angle_mat` and `txrx_network.photon_mat`.

`load_obstacles`: Loads in two lists of obstruction vertices and heights, creates shapes using Python shapely library, and overwrites the values in `obstacles` and `obstacle_heights`.

`create_obstacle_elev_constraints`: Takes a specific communication link as an input and finds which obstacles are in the channel using the shapely library. It then creates new `LinearConstraint` on the elevation angles. The output is the new constraint on both Tx and Rx elevation angles, the obstacles that limit the Tx and Rx nodes, and if the nodes are line-of-sight (LOS).

`create_links`: Takes as an input a number that corresponds to a Tx node in the network. It then creates a list of instances of `txrx_single_link` that represent the input Tx node communicating to every Rx node. Used to create the data matrices with iterating through all possible Tx nodes and storing.

`create_all_links`: Iterates through `create_links` function for every Tx node.

`create_custom_links`: Takes a list of tuples with nodes to create links with as an input and then creates those specific links.

`Xu_experiment`: Uses models from Xu[2] and runs model on all created links. If two nodes are LOS, it uses the LOS model; otherwise, it uses the NLOS model with the constraints from the environment. Uses Xu functions from `txrx_single_link` class.

`opt_Xu_experiment`: Uses the optimization equations from `txrx_single_link` class on the network using the logic of the previous function.

`opt_NLOS_General_sim_geom`: Uses the model from Zuo et al.[3] and runs the steering optimization on all created links. If a link is LOS, uses LOS equation from Xu[2], otherwise, it uses the `opt_NLOS_General_geom` function from `txrx_single_link` to find the photon rate for all created links.

`calc_SNR`: Takes the photon rate matrix, `photon_mat` made with either

`opt_NLOS_General_sim_geom` or `opt_Xu_experiment` and calculates the signal-to-noise ratio (SNR) with the following equation assuming an OOK modulation scheme:

$$SNR_{i,j} = \frac{photon\_mat_{i,j}}{R}\left(\frac{photon\_mat_{i,j} + noise}{R}\right)^{-\frac{1}{2}} \quad . \tag{B-1}$$

Additionally, outputted is the bit error rate (BER) for every Tx/Rx pair as well as the SNR in decibels. BER is calculated as follows[4]:

$$BER_{i,j} = \frac{1}{2}\mathbf{erfc}\left(\frac{\sqrt{SNR_{i,j}}}{2\sqrt{2}}\right) \quad . \tag{B-2}$$

`calc_SINR`: Finds the signal-to-interference-plus-noise (SINR) value for a specific list of connections. The first tuple in the list is the SINR value we want to investigate; the other tuples dictate interfering nodes. Output is the SINR and BER defined according to Eqs. B-1 and B-2 where the value in `noise` is the sum of the values in `noise` and `interference`.

`find_max_rate`: Finds the maximum data rate depending on `SNR_threshold` and `BER_MAX`. Essentially solves Eq. B-1 and Eq. B-2 for R by setting SNR and BER to their minimum and maximum values, respectively. This is done for every Tx/Rx pair in the network and stored in the corresponding internal variables.

B.3 `UVC_Experiments`

This file takes the previous two files and combines them to create final data and store it in an instance of the `txrx_network` class. The functions that are needed to run the model are detailed in the following subsections.

B.3.1 `opt_General_sim_geom`

This function takes as input an instance of `txrx_network`, K_a, K_s^R, and K_s^M. It then runs `create_links` and `opt_NLOS_General_sim_geom` to generate the optimal photon rates and steering angles for every possible communication link in the network. The data is returned as well as stored in `photon_mat` and `angle_mat`. The two matrices are square and the row of the matrix corresponds

[4]Chen G, Xu Z, Ding H, Sadler BM. Path loss modeling and performance tradeoff study for short-range non-line-of-sight ultraviolet communications. Optics Express. 2009;17(5):3929–3940.

to the transmitting node, while the column corresponds to the receiving node. This matrix format is consistent for every matrix in the software unless otherwise specified.

B.3.2 `interference_single_point`

This function takes as an input an instance of `txrx_network`, a tuple of a desired Tx/Rx connection, and a tuple of an interfering Tx/Rx connection, K_a, K_s^R, and K_s^M. It then uses the UV-C Interference Algorithm in Appendix A to find the interference at the desired receiving node.

B.3.3 `create_interference_matrix`

This function runs the `interference_single_point` for all interference scenarios. It then stores the data in a 4D numpy matrix, $\mathcal{P}$. $\mathcal{P}[i, j, k, \ell]$, which corresponds to the scenario where i is optimized to transmit to j and k is optimized to transmit to ℓ. The value $\mathcal{P}[i, j, k, \ell]$ is defined as the interference seen at receiver j from transmitter k given the previous scenario. More visually, the pointing direction of each transceiver in this scenario is $i \rightarrow j, k \rightarrow \ell \Rightarrow \mathcal{P}[i, j, k, \ell] =$ Signal from $k \rightarrow j$. The default interference value is zero and that occurs when the cones of the beamwidth and FOV do not overlap. This is done for every i, j, k, ℓ combination with the following rule: $i \neq j, k \neq \ell, k \neq j, k \neq i, \ell \neq i$.

List of Symbols, Abbreviations, and Acronyms

2D	two-dimensional
3D	three-dimensional
4D	four-dimensional
BER	bit error rate
CAD	computer-aided-design
EM	electromagnetic
FOV	field of view
LED	light-emitting diode
LOS	line-of-sight
LPD	low probability of detection
NLOS	non-line-of-sight
OOK	on-off-keying
PMT	photon multiplier tube
Rx	receiver
S/I	signal-to-interference ratio
SINR	signal-to-interference-plus-noise
SNR	signal-to-noise ratio
Tx	transmitter
UI	user interface
UV	ultraviolet

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	DEVCOM ARL FCDD RLB CI TECH LIB
1 (PDF)	RICE UNIVERSITY J REKO
1 (PDF)	ORAU H ARSLAN
3 (PDF)	DEVCOM ARL FCDD RLA NB T J MOORE F T DAGEFU J KONG