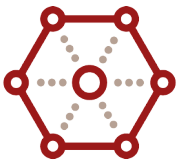
NIST Quasi-Deterministic (Q-D) Channel Realization Software Documentation

In Collaboration with Padova University

by

Anuraag Bodi, Steve Blandino, Neeraj Varshney, Jiayi Zhang, Tanguy Ropitault, Mattia Lecci, Paolo Testolina, Jian Wang, Chiehping Lai and Camillo Gentile



WIRELESS NETWORKS DIVISION, COMMUNICATION TECHNOLOGY LAB

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY,

GAITHERSBURG, MD, USA

JANUARY, 2021

Software Disclaimer

NIST-developed software is provided by NIST as a public service. You may use, copy and distribute copies of the software in any medium, provided that you keep intact this entire notice. You may improve, modify and create derivative works of the software or any portion of the software, and you may copy and distribute such modifications or works. Modified works should carry a notice stating that you changed the software and should note the date and nature of any such change. Please explicitly acknowledge the National Institute of Standards and Technology as the source of the software.

NIST-developed software is expressly provided "AS IS." NIST MAKES NO WARRANTY OF ANY KIND, EXPRESS, IMPLIED, IN FACT OR ARISING BY OPERATION OF LAW, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT AND DATA ACCURACY. NIST NEITHER REPRESENTS NOR WARRANTS THAT THE OPERATION OF THE SOFTWARE WILL BE UNINTERRUPTED OR ERROR-FREE, OR THAT ANY DEFECTS WILL BE CORRECTED. NIST DOES NOT WARRANT OR MAKE ANY REPRESENTATIONS REGARDING THE USE OF THE SOFTWARE OR THE RESULTS THEREOF, INCLUDING BUT NOT LIMITED TO THE CORRECTNESS, ACCURACY, RELIABILITY, OR USEFULNESS OF THE SOFTWARE.

You are solely responsible for determining the appropriateness of using and distributing the software and you assume all risks associated with its use, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and the unavailability or interruption of operation. This software is not intended to be used in any situation where a failure could cause risk of injury or damage to property. The software developed by NIST employees is not subject to copyright protection within the United States.

Contents

Abbreviations

2D Two-Dimensional.
3D Three-Dimensional.
5G Fifth Generation.
6G Sixth Generation.

AMF Additive Manufacturing File Format.
AoA Angle of Arrival.
AoD Angle of Departure.
AP Access Point.

CAD Computer-Aided Design.
CSV Comma-separated values.

DDIR Double Directional Channel Impulse Response.
DoA Direction of Arrival.
DoD Direction of Departure.

GIS Geographic Information System.

JSON JavaScript Object Notation.

LOS Line-Of-Sight.

MIMO Multiple-Input Multiple-Output.
mm-wave millimeter-wave.
MPC Multipath Component.

NIST National Institute of Standards and Technology.

OSM Open Street Map.

PAA Phased Array Antenna.

Q-D Quasi-Deterministic.

Rx Receiver.

STA Station.

TGay 802.11ay Task Group.
Tx Transmitter.

VR Virtual Reality.

XML eXtensible Markup Language.

Notation

τ	Delay
d_{mpc}	Path length
c	Velocity of light
λ_0	Wavelength
f_0	Frequency of the wave
Δf	Change in frequency due to Doppler effect
$\Delta\phi$	Phase shift due to Doppler effect
Δv	Relative velocity between nodes
θ	AoA/AoD elevation angle
ϕ	AoA/AoD azimuth angle
$L_{\text{pre/post}}$	Number of pre/post cursor diffuse components
L	Sixe of the cluster
M	Number of multipath components
N_{PAA}	Number of PAA per transceiver node
N_{node}	Number of nodes
R_{surf}	Number of reflective surfaces
N	Order of reflection
K-factor	The ratio of specular power to the sum of diffuse power
PG	Path gain
γ	Rate of decrease of power of diffuse components with respect to specular component
λ	Delay spread parameter
Θ	Angular spread parameter
s, σ	Noncentrality and scale parameters of Rician distribution
μ, σ^2	Mean and variance of Gaussian distribution
$\mathcal{G}$	Global coordinate system
$\mathcal{N}$	Local coordinate system
R	Rotation of the local coordinate system with respect the global coordinate system

NIST Q-D Channel Realization Software Documentation

This document presents the NIST Q-D Channel Realization Software, which generates realistic multi-point to multi-point millimeter-wave (mm-wave) Double Directional Channel Impulse Response (DDIR), for numerous outdoor and indoor environments, such as an outdoor urban parking lot and an indoor data center, that have been accurately measured and analyzed [1]. The NIST Q-D Channel Realization Software enables the simulation, analysis and test of both physical and link layers of mm-wave wireless communication and sensing systems.

1 Introduction

The NIST Q-D Channel Realization Software is a Three-Dimensional (3D) ray tracing software developed in Matlab[1]. It provides a flexible, scalable and realistic channel model based on measurement campaigns [1] to promote the design of new generation wireless communication and sensing systems at mm-wave bands.

The main features of the NIST Q-D Channel Realization Software can be summarized as follows:

- Support of a flexible scenario configuration with multiple nodes.
- Accurate channel description thanks to geometrical ray tracing dependent on the input Computer-Aided Design (CAD) model of the environment and thanks to diffuse scattering models based on extensive measurements.
- Support of node mobility.
- Description of the propagation in indoor and outdoor environments, using National Institute of Standards and Technology (NIST), 802.11ay Task Group (TGay) or custom material libraries.

[1]Tested with versions R2019b+

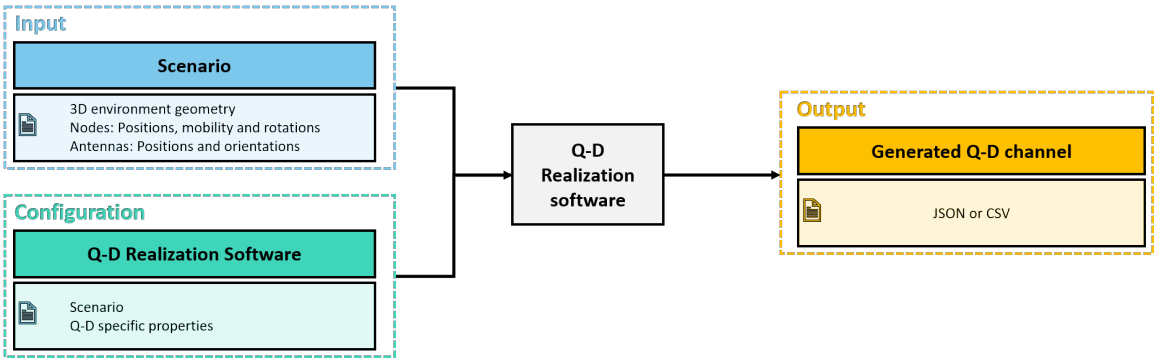


Figure 1: The NIST Q-D Channel Realization Software workflow.

- Support to Multiple-Input Multiple-Output (MIMO) technologies: scaling the propagation description to nodes with multiple antenna arrays overcoming the increased computation complexity.
- Configurable antennas orientation.
- Configurable orientation of each device at discrete points in time.

Figure 1 presents an overview of the NIST Q-D Channel Realization Software workflow. The software generates the mm-wave Double Directional Channel Impulse Response (DDIR) using the Quasi-Deterministic (Q-D) methodology [2], which requires the environmental definition to accurately perform the ray tracing.

The workflow to obtain the DDIR can be summarized in three main steps:

1. The first step is the scenario definition. A scenario consists of a 3D environment geometry and the topology of the network such as nodes position and orientation over time, antennas position and orientation. Readers interested about the creation of a scenario can refer to Section 4.1.1.

   To accurately predict the channel characteristics at mm-wave frequencies, the NIST Q-D Channel Realization Software accounts for the geometry and the materials of the environment. Figure 2 shows one indoor (lecture room) and one outdoor (street canyon) examples. These scenarios are part of the examples scenarios delivered with the NIST Q-D Channel Realization Software (see Section 5), which include also:

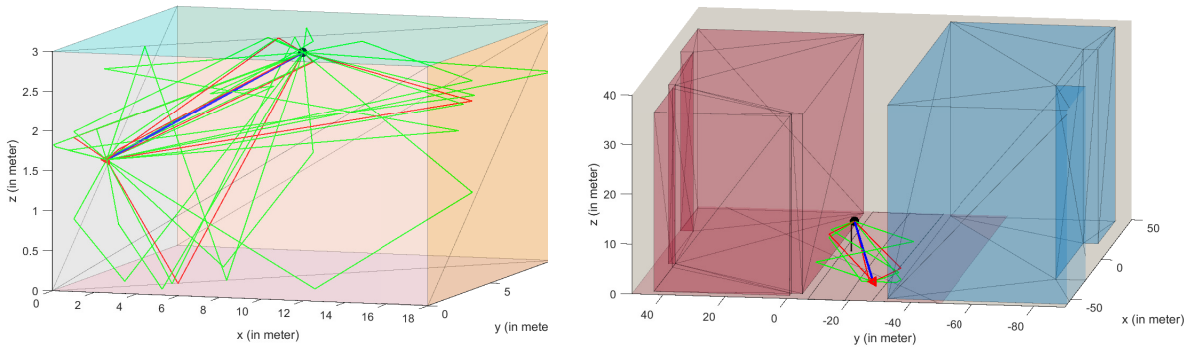


Figure 2: Indoor and outdoor scenarios with two nodes simulated up to second-order reflections. Blue line is the Line-Of-Sight (LOS), the red lines are first order reflections, and the green lines are second order reflections.

- Indoor environments
  (a) Lecture room
  (b) L-shaped room
  (c) Data center
  (d) Living room
  (e) Hotel lobby
- Outdoor environments
  (a) Parking lot i.e., urban downtown
  (b) Street canyon
  (c) Open area hotspot
  (d) City block

The NIST Q-D Channel Realization Software allows a quick and highly Customizable network configuration. For instance, Figure 3 shows an example of indoor dense network, where each of the Virtual Reality (VR) headsets and Access Points (APs) are configured with a different position and orientation.

2. The second step is the configuration of the rays in the Q-D-model such as reflection orders and diffuse components properties. The parameters configuration sets also run-time parameters such as number of time instances generated or sampling time. Section 4.1.2 describes in detail the configuration parameters of the NIST Q-D Channel Realization Software.

   For instance, Figure 2 shows a ray tracing performed up to the second-order reflection between two nodes, which guarantees an accurate channel description.

3. Finally, the last step is the exploitation of the Q-D channel output generated by the NIST Q-D Channel Realization Software. Information about the generated output can be found in Section 4.2.

   Figure 4 shows an example of point-to-point DDIR obtained as output of the NIST Q-D Channel Realization Software. The path gain of each ray is described

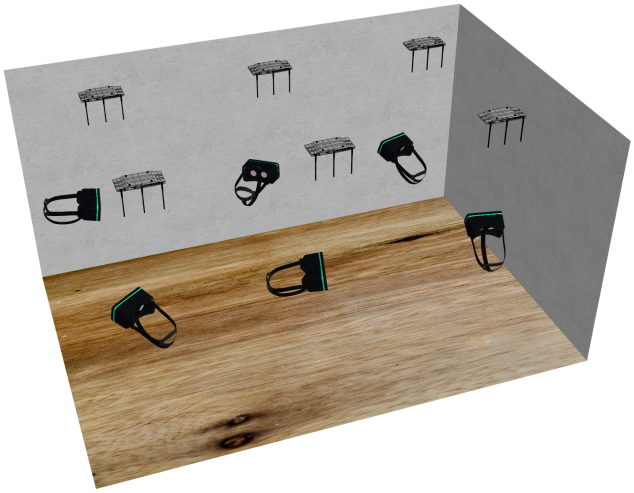


Figure 3: Indoor dense network scenario with multiple VRs and APs.

in time domain. The mm-wave sparsity nature of the channel is shown by the joint representation of Direction of Arrival (DoA) and Direction of Departure (DoD).

The output propagation channel is described for all the possible node-pair combinations enabling the analysis of complex networks such as device to device (D2D) systems or distributed networks.

2 Q-D Model

The mm-wave wireless propagation is modeled as a collection of rays, each representing a separate electromagnetic wavefront. For the design and simulation of current Fifth Generation (5G) and future Sixth Generation (6G) networks, channel models that provide a deterministic description of space and time of the rays are necessary.

The NIST Q-D Channel Realization Software implements geometric ray tracing to capture the deterministic components of all the rays. Since some physical phenomena, e.g. rough surface scattering, cannot be easily modeled in a deterministic manner, NIST has conducted several mm-wave channel measurements campaigns in various outdoor and indoor environments at 28 GHz and 60 GHz, to integrate the deterministic channel description with realistic stochastic models [3]. In addition, NIST implements the Q-D channel model proposed by the IEEE TGay [4] and stochastically models the rough surface scattering using the parameters extracted through rigorous channel measurements carried out by NIST [1] and TGay [2].

The NIST Q-D Channel Realization Software has two major engines to compute the rays between points in the 3D space: a **deterministic** engine and a **stochastic** engine, as described in detail in Section 6.5 and Section 6.1, respectively. Each of the

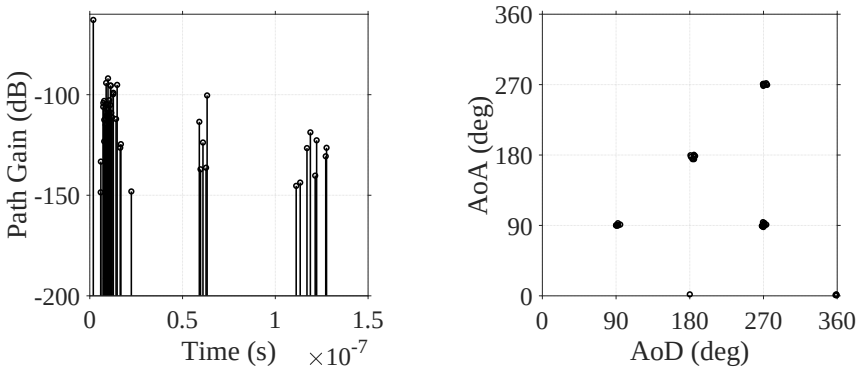


Figure 4: Space-time point-to-point DDIR.

rays is characterized by the delay, the path loss, the phase, the Angle of Arrival (AoA) and Angle of Departure (AoD).

Geometric ray tracing is implemented by extracting the geometrical features of an environment stored in CAD or Geographic Information System (GIS) file formats. The NIST Q-D Channel Realization Software has chosen CAD files to extract geometrical features from a given environment since they can be generated easily for both indoor and outdoor environments using open source CAD software. The deterministic engine generates specular rays between any two points in space. These points in the 3D space approximate the focal point of the antennas of a wireless system, i.e. the points from which the wireless signal is generated or collected. Novel wireless systems include the possibility to include several antennas at each station to enable MIMO operations. For this reason, the NIST Q-D Channel Realization Software has introduced the possibility to generate multi-point to multi-point DDIR, as described in Section 2.2.

Using the specular rays as the basis, diffuse rays are generated from stochastic models, which describe the statistical distribution of reflection loss and rough surface scattering for each of the reflectors present in the environment. The measurements and the analysis of the materials obtained thanks to NIST measurements are collected in *material libraries* described in Section 6.1.1. For the sake of completeness, NIST includes also the stochastic model and the material libraries proposed in the TGay channel document [2] (described in Section 6.1.2) extending the range of supported scenarios. Moreover, users can build customized material libraries to enhance the fidelity of the application to simulate.

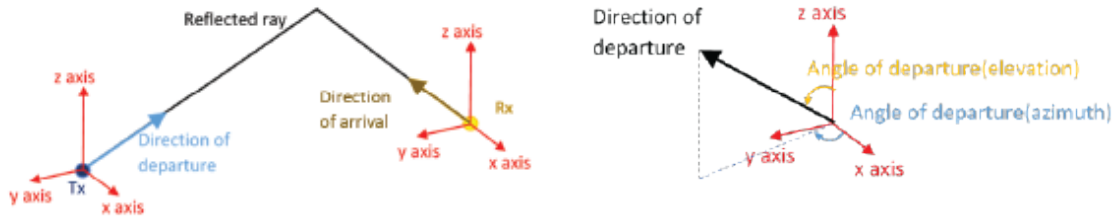


Figure 5: (Left) Reflected ray and its direction of arrival and departure vectors. (Right) The direction of departure is decomposed into azimuth and elevation planes.

2.1 Ray Properties

In the NIST Q-D Channel Realization Software, the mm-wave propagation is modeled as a collection of rays, each representing a separate electromagnetic wave front. The NIST Q-D Channel Realization Software comprises the description of strong deterministic rays (D-rays) obtained using ray tracing. The rays are assumed to propagate in clusters, each one composed by a cluster-head, i.e., a D-Ray, followed by L_{post} stochastic post-cursor rays, and preceded by L_{pre} pre-cursor rays such that the size of the cluster is $L = 1 + L_{\text{pre}} + L_{\text{post}}$. The pre/post cursor rays represent the impact of rough-surfaces as well as the presence of irregular objects on the considered reflecting surface.

Each of the rays is described with its geometrical properties. The main geometrical properties of a single ray between two points are the following:

- Path length, which is defined as the length of the propagation path of the ray between two points, e.g. between a Transmitter (Tx) and a Receiver (Rx) of a wireless communication system.

- The DoA and the DoD in the respective coordinate systems of Rx and Tx, denoting the direction from which the wavefront impinges on the antennas or the direction of the waveform transmission.

The angle of the DoA and DoD vectors are called AoA and AoD respectively. AoA and AoD have two components: the azimuth $\phi \in (0, 360)$ defined in the horizontal plane and the elevation $\theta \in (0, 180)$ defined in the vertical plane. These are the primary attributes of a ray and they are shown in Figure 5. Other ray characteristics can be derived from the geometrical properties of a ray:

- **Delay**: The delay τ is defined as d_{mpc}/c, i.e. the ratio of the path length d_{mpc} (m) and the speed of light c (m/s).

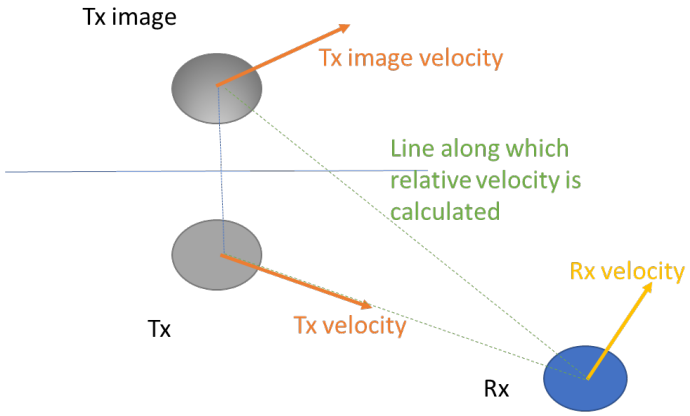


Figure 6: Schematic diagram to calculate the relative velocity of the reflected images for the scenario where two nodes are moving.

- **Free space path loss**: The free space path loss scales with the path length. The path loss in dB can be computed using the Friis transmission loss equation:

$$\text{Path Loss} = 20 \log_{10} \left(\frac{4\pi d_{\text{mpc}}}{\lambda_0} \right), \tag{1}$$

  where λ_0 is the wavelength in meters of the wave.

- **Phase**: The phase of a ray is assumed to be rotated by 180 degrees for every reflection. For example, a second order reflection has a phase shift of 360 degrees. The phase is also affected by the Doppler effect which occurs due to the mobility of the nodes.

- **Doppler effect**: The Doppler effect is frequency shift of the wave perceived by the observer when there is a relative motion between the observer and the source [5]. The phase of the received signal is affected by the relative velocity between two nodes. The frequency shift Δf is given by

$$\Delta f = \frac{\Delta v}{c} f_0, \tag{2}$$

  where Δv is the relative velocity between Rx and Tx, and f_0 is the frequency of the wave. Using the frequency shift in equation (2), the phase shift due to the Doppler effect is given as $\Delta\phi = \Delta f \times \tau$, where $\Delta\phi$ is the phase shift.

  The Doppler effect varies for different reflections because it is dependent on relative velocities of the reflected images but not on the actual node velocities. The relative velocities of the reflected images are calculated in two steps:

  – Compute of the velocities of the individual nodes.

– Find the velocities of the reflected images recursively in the chronological order. The velocity of the node and the velocity of its image are opposite to each other about the normal vector of the reflection plane and constant along the reflection plane as shown in Figure 6. Once the final image is obtained, then the relative velocity is calculated along the line joining the final image and the other node.

2.2 Multi-Point to Multi-Point DDIR

Among different technical advancements, IEEE 802.11ay includes multi-stream transmission that requires devices using multiple transceiver chains. For the design and evaluation of MIMO technologies, channel models need to provide a reliable channel description for each Tx-Rx chain pair. For this reason, the NIST Q-D Channel Realization Software enables the generation of multi-point to multi-point channel description. The NIST Q-D Channel Realization Software describes the channel between points in space, hence the reader can apply any custom antenna model on top of the generated DDIR to obtain the beamformed DDIR for link-level PHY simulations. However, since the usage of Phased Array Antennas (PAAs) is the most common mm-wave MIMO systems implementation [6], these points are named in the following as PAAs.

It is worth noting that the ray tracing computations can be already computationally intensive for a single link. When assuming N_{PAA} PAAs per transceiver node, the number of multipath components M that needs to be described scales as:

$$\begin{aligned} M =& N_{\text{PAA}}^2 \left(1 + \sum_{i=1}^{N} R_{\text{surf}}(R_{\text{surf}} - 1)^{i-1}\right) \binom{N_{\text{node}}}{2} \\ =& N_{\text{PAA}}^2 \frac{R_{\text{surf}}(R_{\text{surf}} - 1)^N - 2}{R_{\text{surf}} - 2} \binom{N_{\text{node}}}{2}, \end{aligned}$$

where N_{node} is the number of nodes, R_{surf} is the number of reflective surfaces, which reflection is described deterministically, and N is the order of reflection. In a large MIMO scenario, where there may be multiple nodes equipped with several PAAs, computing the ray tracing for each link would be computationally expensive.

The NIST Q-D Channel Realization Software implements a scalable multi-PAAss Q-D channel model implementation based on the assumption that PAAs separated by a distance smaller than $d_{\text{fc}} = \lambda/2 \times n_{\text{fc}}$ experience full correlated channels. Here, n_{fc} is the number of half-wavelength the channel is assumed fully correlated. The parameter d_{fc} depend on the environment, the PAA beamwidth and the system bandwidth. For instance, NIST internal measurements show that outdoor environments are more coherent than indoor environments due to less rich scattering [7].

Under these assumptions, we introduce a PAA grouping algorithm that groups the PAA defined by the user, according to the PAAs mutual distance, i.e. the distance between each PAA pair in a node. We identify two cases:

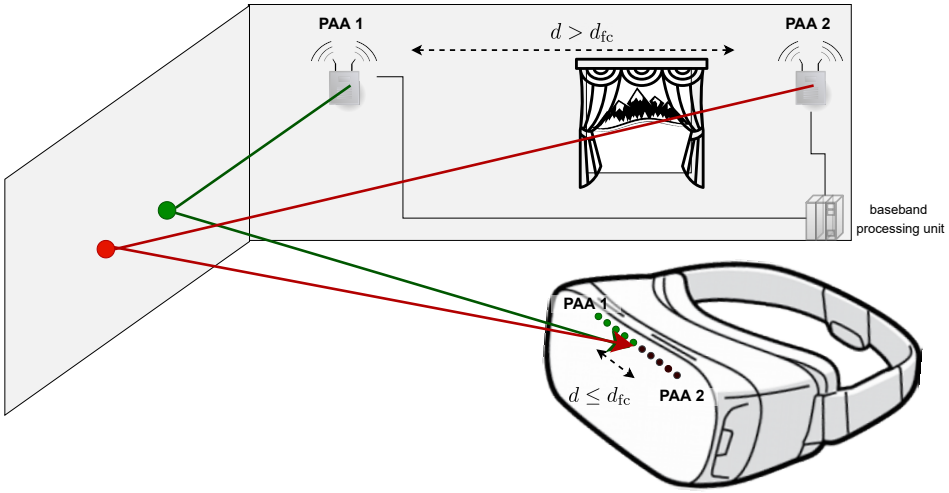


Figure 7: Different cases of multi-PAA devices.

- PAAs are spaced by a distance $d \leq d_{\mathrm{fc}}$. In the the far field of the transmitter antenna and any ambient scatterers, the signals at each PAA only differ by a phase rotation depending on the directions of departure/arrival. In this case, a single channel realization is computed for all PAAs.

- PAAs spaced by a distance $d > d_{\mathrm{fc}}$. In this scenario the channel is weakly correlated, thus both specular and stochastic channel components are independently generated for each Tx-Rx PAA pair.

Application Example: Indoor Distributed mm-wave MIMO Network

Let us assume an indoor single user MIMO wireless network as depicted in Figure 7 to demonstrate how the Q-D software is able to simulate large MIMO systems without a significant run-time increase with respect to a single transmit and receive PAA case.

The AP is equipped with a distributed antenna array, e.g. with $M_{\mathrm{RF}}^{\mathrm{AP}} = 2$ antenna sub-arrays that are distributively located, i.e. $d > d_{\mathrm{fc}}$, such that the sub-arrays are connected to a baseband processing unit via optical fiber. The Station (STA) is instead equipped with $M_{\mathrm{RF}}^{\mathrm{STA}} = 2$ PAAs such that the distance between the center of each array is below the correlation distance d_{fc}. For MIMO simulations, the ray tracing needs to provide the description of each channel from each transmitting PAA of the MIMO AP to each receiver PAA of the MIMO STA. In this scenario the digital MIMO matrix $\mathbf{H} \in \mathbb{C}^{M_{\mathrm{RF}}^{\mathrm{STA}} \times M_{\mathrm{RF}}^{\mathrm{AP}}}$ counts a total of 4 channels, however with the proposed implementation only 2 rays need to be ray traced, i.e. the channels relatives to the PAAs spaced apart more than $d > d_{\mathrm{fc}}$. The other channels corresponding to PAAs spaced apart less than d_{fc} will be obtained with a correction of the phase information.

2.3 Orientation

Beyond moving at a pedestrian speed, mobile nodes change their orientation continuously over time. For instance, VR headsets are subject to sudden change of orientation since users rotate their neck when experiencing multimedia contents. Since antennas create directional radiation patterns, it is important to model the position and rotation of the device, as well as the orientation of the antenna in each device.

To manage the orientation of the device, two coordinate systems are introduced.

- The **global coordinate system**, $\mathcal{G}$, which is fixed to the earth and it is aligned with its x-axis pointing east, its y-axis pointing north, and its z-axis pointing up.
- The **local coordinate system**, $\mathcal{N}$, is centered in the node and moves with it.

Both $\mathcal{G}$ and $\mathcal{N}$ coordinate systems are related via a linear transformation, which can be described mathematically as:

$$p^{\mathcal{G}} = Rp^{\mathcal{N}} + t, \tag{3}$$

where $p^{\mathcal{G}} \in \mathbb{R}^3$ represents a point in the global coordinates, $p^{\mathcal{N}} \in \mathbb{R}^3$ is a point in the node coordinates and $t \in \mathbb{R}^3$ is the translation vector between the origin of the coordinate frames. In practice, t is the position of the node in the global coordinate system and $R \in \mathbb{R}^{3\times3}$ represents the rotation of the local coordinate system with respect the global coordinate system. The rotation R describes how to rotate the point $p^{\mathcal{N}}$ in the local coordinate, to describe it as the point $p^{\mathcal{G}}$ in the global coordinate system. The rotation R can also seen as the rotation that is applied to the global frame to align it with the local frame. The implementation of the orientation in the software is enabled by means of quaternions [8].

2.3.1 Point Rotation vs Frame Rotation

To properly model the interaction between antennas and environment, it is important to describe the orientation of the PAAs. First, the rotation of the device brings the antennas in a different position in the global frame. This rotation will be referred in the following as **point rotation**. Secondly, the rotated PAAs experience different incoming/departing beam angles i.e. AoA and AoD. Therefore AoA and AoD have to be rotated to the local coordinate systems as in the NIST Q-D Channel Realization Software they have been computed with respect to the global frame, before applying the beamforming vectors. In the following, this rotation will referred to as **frame rotation**.

Figure 8 shows the two different rotations. Assuming $\mathbf{p}_s$ the centroid position of the PAA s and $\mathbf{c}$ the centroid position of the device, both in the global coordinates $\mathcal{G}$, the PAA centroid coordinates after the point rotation of the device can be described as:

$$\hat{\mathbf{p}}_s = \mathbf{c} + \mathbf{R}(u)(\mathbf{p}_s - \mathbf{c}). \tag{4}$$

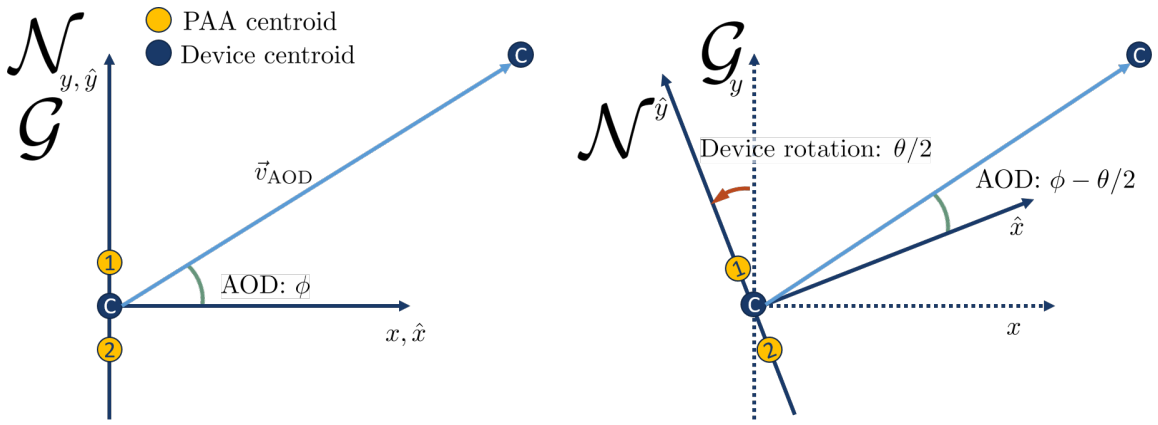


Figure 8: The point rotation is used to model the device rotation, which brings the PAAs in a different position in the global frame. The frame rotation is used to transform AoA and AoD from global to local coordinates.

Frame rotation is the inverse of the point rotation. In frame rotation, the points stay fixed in the frame $\mathcal{G}$, but the local frame of reference $\mathcal{N}$ is rotated. Assuming the AoD being described by the vector $\vec{v}_{\text{AOD}} = \mathbf{c}_{\text{rx}} - \mathbf{c}_{\text{tx}}$, the AOD vector after the frame rotation can be described as:

$$\hat{\vec{v}}_{\text{AOD}} = \mathbf{R}^{-1}(u)\vec{v}_{\text{AOD}}. \tag{5}$$

2.3.2 Consecutive Rotations

In mm-wave, due to the limited antenna field of view, several PAAs might be used to cover a given area. For instance an AP can use different PAAs, each one oriented towards a different sector in space by a rotation $\mathbf{R}_{\text{PAA}}$. Hence, for dynamic devices, e.g. a moving user with a VR headset, two rotations need to be considered. The first rotation is the rotation of the device $\mathbf{R}_{\text{D}}$ where the PAAs are placed, e.g. due to neck rotation, and secondly the rotation of the PAA $\mathbf{R}_{\text{PAA}}$ with respect to the device itself, i.e. the initial orientation of the PAAs. In case of double rotation, the rotation matrix in equations (4) and (5) is written as:

$$\mathbf{R} = \mathbf{R}_{\text{PAA}}\mathbf{R}_{\text{D}}. \tag{6}$$

2.4 NIST vs TGAY

The NIST Q-D Channel Realization Software proposes a Q-D channel model implementation including both NIST and TGay models. While they are both based on deterministic ray tracing, there are conceptual differences.

2.4.1 Diffuse Components

In the NIST model, the diffuse components of a cluster are generated stochastically, thanks to distributions that characterize the path gains, the angular description and the time of arrival of each ray. Each of the parameters that characterizes the cluster, such as K-factor, power decay constant (γ), Poisson arrival rate (λ), are randomly generated using the Rician distribution with noncentrality parameter s and scale parameter σ for each of the reflectors present in the scenario. Note that the values (s, σ) to obtain K-factor, γ, and λ are derived after classifying the Multipath Component (MPC) clusters per reflector. Instead, TGay model proposes stochastic distribution characterized by constant parameters, such as K-factor, γ, and λ. In contrast to NIST model, these parameters are fixed for each of the reflectors present in the scenario. Note that K-factor, γ, and λ parameters are obtained using indoor and outdoor measurements given in the TGay channel document [2].

Moreover, in NIST model, the angular spread of diffuse component in both elevation and azimuth planes follows a Laplacian distribution with zero mean and variance Θ^2, where θ is randomly generated using the Rician distribution. However, in TGay based model, angular spread of diffuse component in both elevation and azimuth planes are generated as independent Gaussian distributed random variables with mean μ and variance σ^2, where μ is equal to the corresponding angle of the specular ray and the σ values for various indoor and outdoor scenarios are given in [2].

2.4.2 Reflection Loss

In the NIST Q-D model, the reflection loss is obtained stochastically thanks to the the measurements campaigns conducted by NIST. Each material reflection loss has been modeled as a Rician stochastic distribution defined by a noncentrality parameter s and a scale parameter σ. Moreover, NIST allows to generate a deterministic DDIR, disabling the diffuse components. In this case, the reflection loss of the specular rays is assumed to be deterministic rather than being extracted from a distribution, and its value is set to the mean of the modeled Rician distribution.

In the TGay the computation of the reflection loss is always deterministic and does not depend on diffuse components are enabled or not. The refection loss for each specular ray is computed thanks to the Fresnel equation, depending on the incident angle(s), the polarization and the material properties. More specifically, for each refection, the NIST implementation of TGay model first considers the reflectances for vertical and horizontal polarization using the incident angle and relative permittivity of the material in the Fresnel equation. Subsequently, the reflection loss for each reflection is calculated by the taking the average of the vertical and horizontal reflectances.

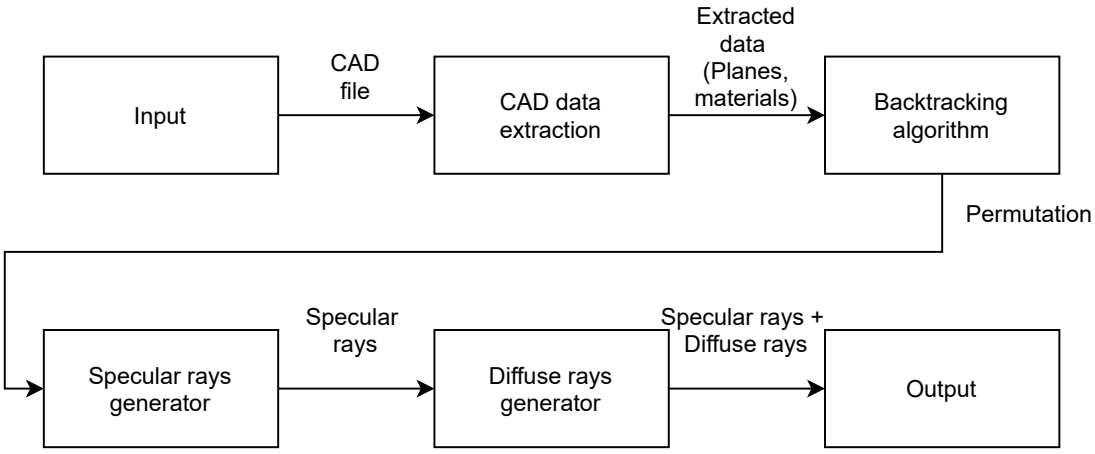


Figure 9: Step-by-step procedure of the NIST Q-D Channel Realization Software.

3 Software Structure

3.1 Block Diagram of NIST Q-D Channel Realization Software

The NIST Q-D Channel Realization Software has been developed to combine the deterministic results computed by ray tracing with the stochastic model obtained from measurements campaigns conducted by the NIST [1] and TGay [2]. Figure 9 gives an overview of the NIST Q-D Channel Realization Software implementation. While each implementation block is described in detail in Section 6, a summary of each step is given in the following list.

- Input: The user has the flexibility to design a specific propagation scenario by defining the input parameters among which the propagation environment, node position and orientation. A set of examples are also provided in Section 5 to guide the user in the configuration. The input is further described in Section 4.1.
- CAD data extraction: From a CAD file the surfaces of the environment, represented as triangle are extracted and described as *planes*. The detailed information about the CAD data file extraction are in Section 6.3.
- Backtracking algorithm: Rather than performing ray tracing on every possible combination of reflecting surfaces, a backtracking algorithm is used to select the permutations of triangles (obtained from the CAD model) combinations, where there is a possibility of reflection. The detailed algorithm is described in Section 6.4.
- Deterministic rays generator: The method of images [9] is used to trace each individual deterministic ray, as described in Section 6.5.

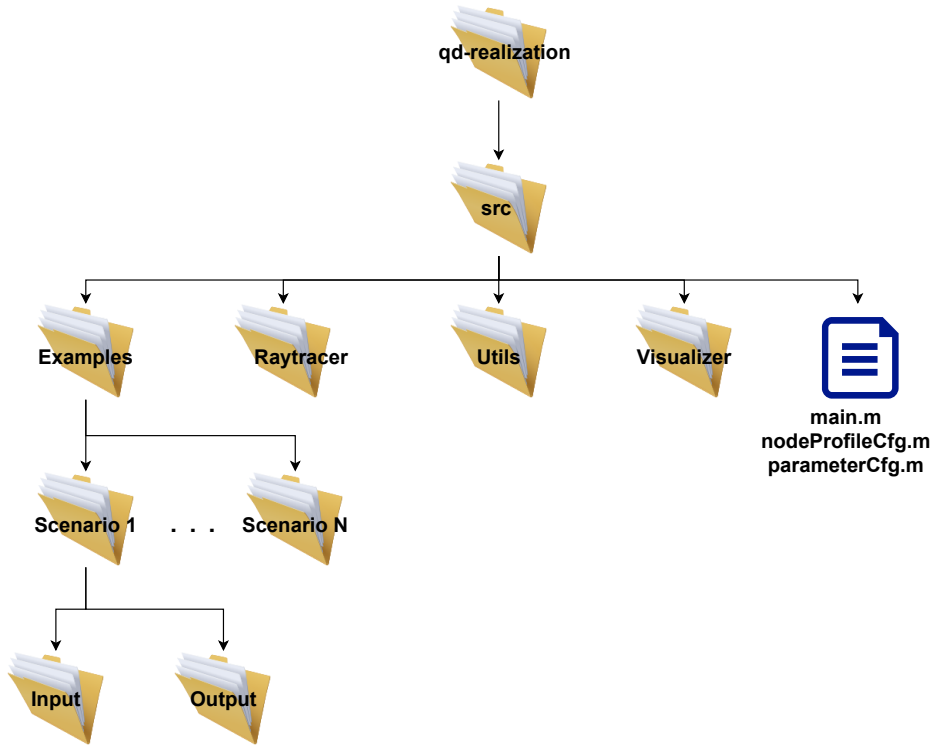


Figure 10: Directory structure of the NIST Q-D Channel Realization Software.

- Diffuse rays generator: The stochastic models are applied on the deterministic ray to generate the cluster. The detailed information can be found in Section 6.1.
- Output: The final output is the DDIR described with the the ray's attributes namely path loss, delay, phase, AoA, AoD at every time step and or antenna combination. Information to visualize the scenario simulated and the ray traced are also given as an output. Please refer to Section 4.2 for detailed information.

3.2 Folder and Code Structure

The following subsections describe the directory and code structures of the NIST Q-D Channel Realization Software.

3.2.1 Directory Structure

The directory structure is shown in Figure 10. The matlab code is collected in the folder `qd-realization\src` containing the main script `main.m` and two parameter configuration functions, i.e., `parameterCfg.m` and `nodeProfileCfg.m`. The folder `qd-realization\src` includes four sub-folders:

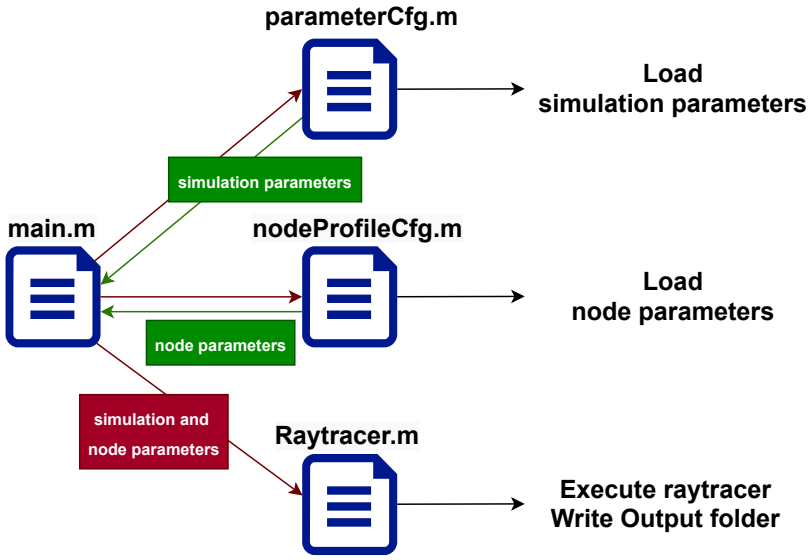


Figure 11: Main components of the NIST Q-D Channel Realization Software.

1. `Examples`: it collects predefined configurations for various indoor and outdoor scenarios, as described in detail in Section 5. Each of the examples has an `Input` folder defining the scenario properties and an `Output` folder that is generated by the Q-D-realization software with the result of the ray tracing.

2. `Raytracer`: it collects all the functions used during the ray tracing execution.

3. `Utils`: it collects helper functions used in the code.

4. `Visualizer`: it collects the functions for the visualizer app[2].

3.2.2 Code Structure

The NIST Q-D Channel Realization Software uses four major files namely the script `main.m`, the functions `parameterCfg.m` and `nodeProfileCfg.m` in the main-folder `qd-realization\src`, and the function `Raytracer.m` in sub-folder `Raytracer`.

Their dependency is illustrated in Figure 11. The script `main.m` invokes the `parameterCfg.m` function to import the scenario input parameters such as number of time samples and the order of reflection from files present in the `Input` folder and return them to the main file. The `main.m` also calls the `nodeProfileCfg` function to import the node profile information such as the node positions and orientations over time from the files defined in the `Input` folder and return them to the `main.m` script. The `main.m` script passes all the above mentioned input parameters as an argument of the `Raytracer.m` function to generate the DDIR.

[2]to be released in a subsequent deliverable

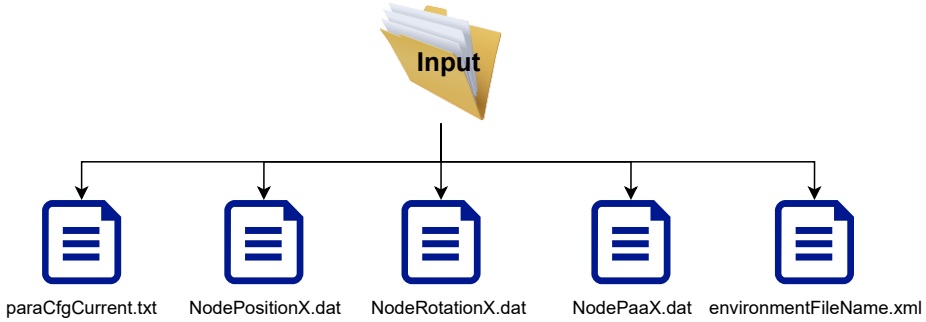


Figure 12: Files in `Input` folder.

4 Usage Description

This section describes how to setup the NIST Q-D Channel Realization Software and how to analyze the results obtained.

4.1 Input

To execute the NIST Q-D Channel Realization Software, the input must be properly defined in the folder `Input` as shown in Figure 12. The input configuration consists of two steps:

- The creation of the scenario (Section 4.1.1) defined by the files:
  - `environmentFileName.xml`
  - `NodePositionX.dat`
  - `NodeRotationX.dat` (optional)
  - `NodePaaX.dat` (optional)
- The configuration to apply to the NIST Q-D Channel Realization Software to generate the scenario's DDIR (Section 4.1.2), defined by the file:
  - `paraCfgCurrent.txt`

4.1.1 Scenario Definition

The NIST Q-D Channel Realization Software relies on ray tracing to obtain the specular components. Thus, each scenario must include a CAD file to describe the 3D geometrical information of the scenario's environment. These geometrical properties can be represented as point clouds, lines, and/or polygons. In the NIST Q-D Channel Realization Software, Additive Manufacturing File Format (AMF) file format is used as the standard-form of input CAD file format, which represents geometrical structures

as triangles. In addition to geometrical properties, AMF file, which can be generated using CAD software[3] (e.g., AutoCAD, FreeCad, etc.), can store material properties of each triangle. Note that this file format is an eXtensible Markup Language (XML) based format and for a detailed description, please refer to [10]. Moreover, examples of AMF files can be found in the `src/Examples` folder.

Each scenario must additionally define the devices properties. The devices are the source and destination points used by the NIST Q-D Channel Realization Software to perform ray tracing. Devices can be defined with two layers of granularity:

- Nodes: The devices itself. It includes the devices position (mandatory), and orientation (optional).
- The Devices Antennas (optional). It includes the antennas coordinates, and orientation relative to the device they belong. If the antennas are defined for one device, antennas positions are used as source and destination of the ray tracing, else, the nodes positions are used.

The nodes position evolution over time is defined using the `NodePosition`X`.dat` file. For each node of the scenario, `NodePosition`X`.dat` needs to be defined. `NodePosition`X`.dat` is using the following format: the position of each node are specified as a 1-by-3 real-valued vector, being $X = 0 \ldots$ `numberOfNodes`-1 the node index. Each row n of the file represents the position of the node (x_n, y_n, z_n) expressed in meters at the time sample $(n-1)T_s$, where T_s denote the sampling time interval. For instance a movement of 0.50 m along the x-axis of node 1 can be configured by defining the file `NodePosition1.dat` as:

...
1,1,3.5
1.5,1,3.5
...

Note that for static node, the number of rows in `NodePosition`X`.dat` should be one. For mobility over time, `NodePosition`X`.dat` should have n rows for n time-instants. In case this file has more rows than n time-instants, then only first n rows would be processed.

The rotation of the nodes over time is defined using the `NodeRotation`X`.dat` file. This file is optional. The node orientation, in radians, is specified as a 1-by-3 real-valued vector. Each element specifies the angle by which the local coordinate system of the node is rotated with respect to an axis of the global Cartesian coordinate system, following the zxy Tait–Bryan convention: the first element is the angle of rotation about the z-axis; the second element is the angle of rotation about the rotated x-axis;

[3]The identification of any commercial and open-source product or trade name does not imply endorsement or recommendation by the National Institute of Standards and Technology, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

the third element is the angle of rotation about the rotated y-axis. A positive value indicates a clockwise rotation. Each row n of the file represents the orientation of the node at the time sample $(n-1)T_s$. For instance an AP mounted on the ceiling pointing down needs to be rotated by $\pi/2$ rad about the x-axis, hence assuming the AP is node 0, the file `NodeRotation0.dat` needs to be configured as:
0,0,1.57
For static orientation, the number of rows in `NodeRotation`X`.dat` should be one. If we have different orientation over time, `NodeRotation`X`.dat` file should have n rows for n time-instants. In case this file has more rows than n time-instants, then only first n rows would be processed. Since this feature is optional, `NodeRotation`X`.dat` file can be omitted and in that case the software does not consider device rotation, i.e. the node frame is assumed to be parallel to the global frame.

Finally, the PAA coordinates and orientation are defined in `NodePaa`X`.dat`. This file is also optional. It defines the position of each PAA of the node X with respect to the center of the node X and their orientation with respect to the local frame of the node X. It is configured as a 1-by-6 real valued vector in which the first 3 values correspond to the distance vector from the center of the node and the last 3 values correspond to the PAA orientation (same angle definition as in `NodeRotation`X`.dat` configuration). Each row r of the file represents a different PAA. Note that the values of this configuration file are not time dependent, i.e. only one set of values should be provided per PAA as the PAAs are assumed to keep the same orientation in the frame of the node. For instance, assuming a device with 360° coverage on the horizontal plane mounting two PAAs covering half of the plane. The `NodePaa0.dat` will be configured as follow:
0, 0, 0, 0, 0, 3.14
0, 0, 0, 0, 0, 0
In case the file `NodePaa`X`.dat` is omitted a single PAA oriented towards the x-axis is assumed, i.e. with local coordinates aligned to the global coordinates.

4.1.2 Configurable Parameters

After the creation of the scenario, the software can be configured to apply the desired models and Q-D parameters to return the DDIR relative to the scenario. The `parameterCfg.m` function imports the configuration parameters listed in the configuration file `Input/paraCfgCurrent.txt`:

1. `environmentFileName` - It defines the CAD file name of the environment in which the ray tracing is executed as specified in Section 4.1.1.

2. `indoorSwitch` - This switch defines whether the scenario is purely indoor or outdoor. For indoor scenario, `indoorSwitch` should be set as 1. Whereas for outdoor scenario, `indoorSwitch` should be set as 0. The default setting in `parameterCfg.m` is `indoorSwitch` = 1 in case this switch is missing.

3. `totalTimeDuration` - This parameter defines the total time evolution in seconds of the scenario. It influences the rate at which each channel instance is generated since T_s = `totalTimeDuration`/`numberOfTimeDivisions`, being `numberOfTimeDivisions` the number of channel realizations in time domain (see below). The default setting is `totalTimeDuration` = 1.

4. `numberOfTimeDivisions` - This parameter defines the total number of time steps, i.e. the number of time samples to simulate. If `numberOfTimeDivisions` = 100 and `totalTimeDuration` = 10, then we have 100 time samples for 10 seconds, i.e. the channel is simulated with a sampling time of 0.1s.

   Note that for static node, `numberOfTimeDivisions` should be 1 since the `NodePosition`X`.dat` file has one row. For mobile case, `numberOfTimeDivisions` can be either lesser or equal to the number of rows in `NodePosition`X`.dat`. In former case, only first `numberOfTimeDivisions` rows in `NodePosition`X`.dat` will be processed by the software. The same conditions are applied on the `NodeRotation`X`.dat` file.

5. `referencePoint` and `selectPlanesByDist` - These parameters are useful to reduce the time-complexity of the software when the nodes are not far away in a very large scenario e.g. a city block with several buildings. The parameter `referencePoint` defines the center of the limiting sphere[4], which should be close to the nodes. Moreover, `selectPlanesByDist` defines the radius of the limiting sphere. The default setting for `referencePoint` and `selectPlanesByDist` are considered as [0,0,0] and `inf`, respectivily. Note that `selectPlanesByDist` = `inf` means that there is no limitation.

6. `switchDiffuseComponent` - This parameter indicates whether the diffuse component is on or not. If `switchDiffuseComponent` = 1 then the diffuse or intra-cluster components are generated along with the specular components, otherwise only the specular components are generated. The default setting is considered as `switchDiffuseComponent` = 0 in case this parameter is missing in `parameterCfg.m`.

7. `diffusePathGainThreshold` - This parameter is employed to filter out some of the diffuse components generated for each cluster. For example, if `diffusePathGainThreshold` is set as -25 dB that means the software considers only diffuse components up to 25 dB below the specular component path gain. The default value is set as `-inf` which means software does not discard any diffuse components. Note that this switch is only used when `switchDiffuseComponent` = 1 for NIST measurement based scenarios, i.e., `switchQDModel` = `nistMeasurements`.

[4]The limiting sphere delimits the area, in which the nodes must be located, to be considered for raytracing

8. `switchQDModel` - This parameter is used to toggle between the two different Q-D models that are used to calculate the reflection loss and diffuse components for each specular ray, i.e. NIST or TGay. In `parameterCfg.m` file, the `switchQDModel` parameter value should be either `nistMeasurements` for NIST measurement based scenarios or `tgayMeasurements` for 802.11 ay channel document [2] based scenarios. In case this switch is missing, the default value is `nistMeasurements`.

9. `materialLibraryPath` - This parameter describes the location of the material library file. Note that this software uses NIST material libraries to obtain reflection loss and diffuse components for NIST measurement based scenarios. For example, for the NIST lecture room scenario, `materialLibraryPath` is `material_libraries/materialLibraryLectureRoom.csv`.

   On the other hand, for 802.11 ay channel document [2] based scenarios, this software uses `intraClusterTgayParameters.txt` file to generate diffuse components and material library file defined by `materialLibraryPath` parameter to calculate the reflection loss.

   If `materialLibraryPath` parameter field in `parameterCfg.m` file is missing, the software displays a warning 'Material library path not defined. Using Empty material library'. In this case the software automatically disables the diffuse components and considers a fixed reflection loss value (see `reflectionLoss` parameter below) for each refection of specular ray.

10. `reflectionLoss` - In case the material library file does not exist or `materialLibraryPath` in `parameterCfg.m` file is not defined or material library file format is not correct for a considered `switchQDModel`, the reflection loss is assumed to be independent of the environment and set to the constant value `reflectionLoss` defined in `parameterCfg.m` file. In case this parameter is missing, the default value of 10 dB per reflection order is considered.

11. `totalNumberOfReflections` - This parameter denotes the highest order of reflection considered in the ray tracer. For example, `totalNumberOfReflections` = 2 means the model considers the LOS, the first and the second-order reflections. The default setting is `totalNumberOfReflections` = 2.

    Further note that for `switchQDModel` = `nistMeasurements`, user can generate rays considering `totalNumberOfReflections`>2. However, for `switchQDModel` = `tgayMeasurements`, `totalNumberOfReflections` can not be considered higher than the default value (i.e., 2) since angles of incident in this software are obtained till second order reflections. In case user defines `totalNumberOfReflections`>2 with `switchQDModel` = `tgayMeasurements`, software would generate rays up to second order reflections.

12. `switchSaveVisualizerFiles` - It allows exporting files to visualize scenario with multipath components in visualizer if `switchSaveVisualizerFiles` = 1. The default setting is considered as `switchSaveVisualizerFiles` = 0 in case this parameter is missing in `parameterCfg.m`.

13. `carrierFrequency` - This parameter defines the carrier frequency in Hz. Note that the default value is considered as 60 GHz in case this parameter field is missing in `parameterCfg.m` file.

14. `qdFilesFloatPrecision` - It defines the number of digits after comma in the output files. The default value is set as 6.

15. `outputFormat` - It defines the format of the output files generated by the software. The current version supports *.json and *.txt file formats. Therefore, the `outputFormat` parameter can assume the following values: `json`, `txt` or `both`.

16. `useOptimizedOutputToFile` - It optimizes write to file when set to 1. Set to 0 if Matlab returns 'Too many files open' error. This parameter is relevant only if `outputFormat` is set to `txt` or `both`.

4.2 Output

NIST Q-D Channel Realization Software provides as output the details of the multi-point to multi-point DDIR, i.e. the magnitude, phase, time of arrival, DoD and DoA of individual propagation paths. The NIST Q-D Channel Realization Software can generate also information to visualize the ray tracing.

The Q-D output can be provided in two file formats: JavaScript Object Notation (JSON) or Comma-separated values (CSV). The JSON output reduces the number of output files and it reduces the execution time since the output file is written only once at the end of the ray tracing operations. Instead CSV files are written at run-time, i.e. at each time sample a new file is opened, written and closed. While improving execution time, JSON output might be heavy on RAM as it will retain data in memory during ray tracing.

While both formats are supported for back compatibility with previous versions of the software, the latest features such as multi-point to multi-point DDIR, node and antenna orientation are only supported in JSON format.

The visualizer output is provided only in JSON format.

4.2.1 CSV Output

The output consists of ray's attributes namely path loss, delay, phase, AoA, AoD at every time-step and Tx Rx node combination. Output files are generated and saved in the `Output` folder which contains two sub-folders: `Ns3` and `Visualizer`, as shown in Figure 13.

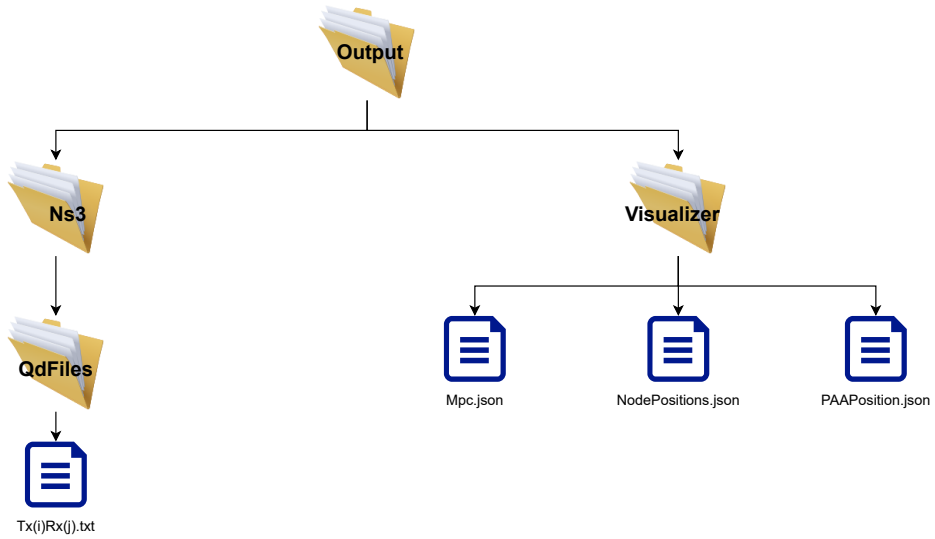


Figure 13: File structure of `Output`.

`Ns3` further contains two sub-folders: `NodesPosition` and `QdFiles`, which are described in detail below.

1. `NodesPositions` - this folder consists of `NodesPosition.txt` file. The file contains the node positions in meters at the first time interval.

2. `Qdfiles` - this folder consists of `Tx(i-1)Rx(j-1).txt` trace files for $(i-1)$th and $(j-1)$th nodes. The format of this file for the scenario when the nodes are stationary is given in following order.

   - Number of rays.
   - Delay of each ray in seconds.
   - Path gain of each ray in dB
   - Phase offset of each ray in radians
   - AoD (elevation) of each ray in degree
   - AoD (azimuth) of each ray in degree
   - AoA (elevation) of each ray in degree
   - AoA (azimuth) of each ray in degree

   The above format is the same for each time instances. However, this format is repeated for each time instances if the nodes are mobile. For example, the first time instance data is stored in the first eight rows, the next eight rows stores second time instance data and so on.

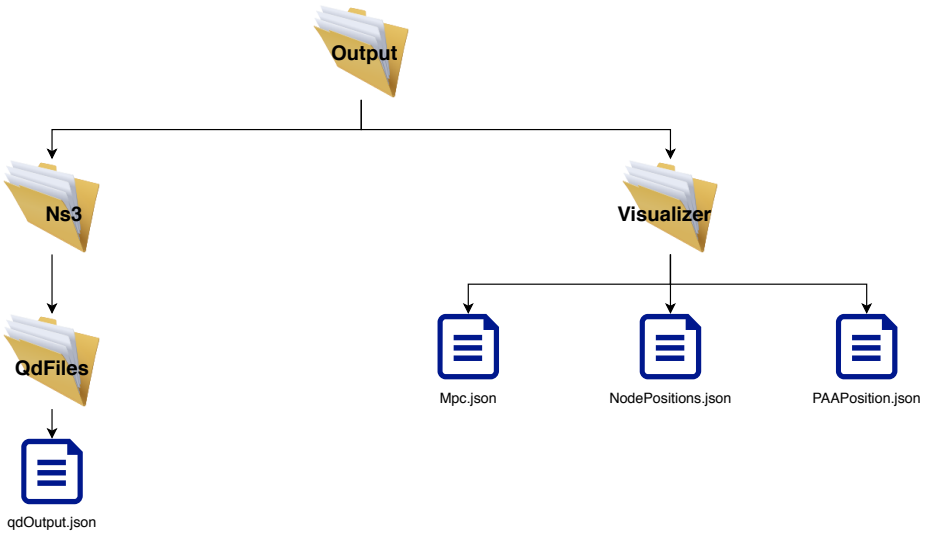


Figure 14: Output JSON file structure.

For a single time instance, the multi-point to multi-point DDIR can be also returned. In this case, the 8 output variables are repeated for each source-destination point, thanks to a loop on the source points, followed by a loop on the destination points. For instance, assuming 2 nodes (Tx, Rx) with 2 PAAs each (1,2) the parameter list is returned in the following order: Tx_1-Rx_1, Tx_1-Rx_2, Tx_2-Rx_1, Tx_2-Rx_2 The dynamic multi-point to multi-point DDIR is supported only with JSON output.

4.2.2 JSON Output

Using the JSON output format, the number of files in output are greatly reduced. Figure 14 shows the output folder structure, when JSON output is used.

The NS3 folder includes the file `qdOutput.json` in the folder `QDFiles.json`. `qdOutput.json`, represented in Figure 15, includes the channel information. Each row of the file describes a combination transmitter-transmitter PAA-receiver- receiver PAA. Each row includes the following field: `Delay` (s), `Gain` (dB), `Phase` (rad), `AOD EL` (deg), `AOD AZ`(deg), `AOA EL`(deg), `AOA AZ`(deg). Each of these fields is a vector of length `numberOfTimeDivision`, hence describing the evolution of each parameter over time. Moreover, each time sample entry is a vector describing the properties at each MPC.

The visualizer folder includes the file `MPC.json`, `NodePosition.json` and `PAAposition.json` and they are represented in Figure 16.

`MPC.json` includes the ray-bouncing points in space. Each row describes the MPC trajectory for a combination transmitter-transmitter PAA-receiver- receiver PAA and for a reflection order. Each row includes the MPC field which is a vector of length

`numberOfTimeDivision`. Each time sample entry includes the 3D coordinates of the intersection points.

Each row of `NodePosition.json` returns the spatio-temporal evolution of a node. It includes the field `Position` and `Rotation`, which are vectors of size `numberOfTimeDivision`. The n-th time sample entry of `Position` is the 3D coordinates of the node, while `Rotation` includes the three rotation angles ordered as ZXY (cfr Section 4.1 for angles definition).

Finally `PAAposition.json` includes information relative to each PAA of each node. The `centroid` field returns information about the clustering: each PAA has been assigned to a cluster of PAA and the ray tracer has been executed for the centroid of the cluster. `centroid` returns the index of the centroid for which the ray tracing has been executed to obtain the channel relative to `PAA`. The field `Orientation` is obtained from the PAA input configuration and it is time independent. The field `Position` is a vector of length `numberOfTimeDivision`, which describes the PAA position in the global frame.

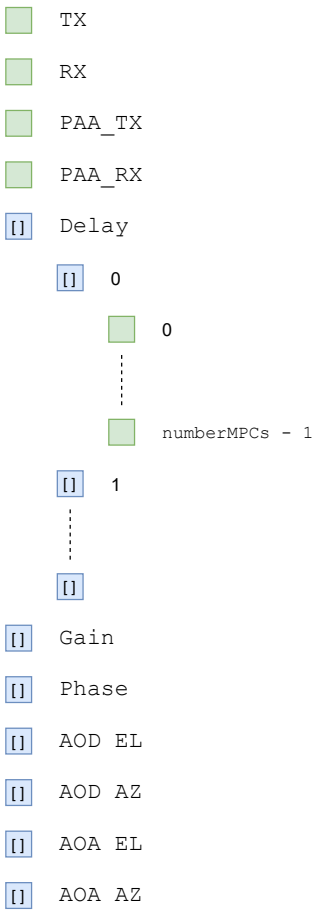


Figure 15: NS3 JSON file.

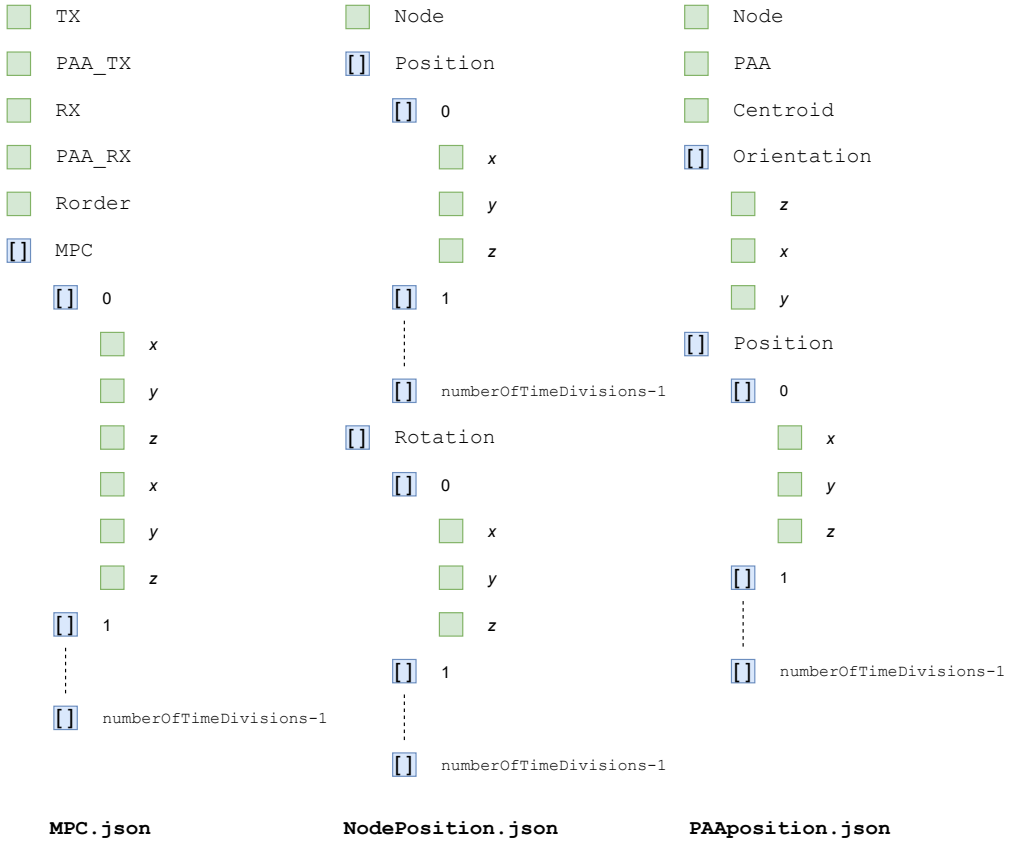


Figure 16: Visualizer JSON files.

5 Scenarios

This section presents several indoor and outdoor example scenarios to guide the user through the available features of the NIST Q-D Channel Realization Software. For better understanding, Table 1 describes all the examples scenarios that are predefined for the user. More specifically, these predefined scenarios are given as examples in `/src/examples/`, where `paraCfgCurrent.txt` file in `Input` folder has been configured to generate the output in corresponding `Output` folder. Users can use these scenarios to start their own simulation design customizing node location, rotation and the other input parameters.

It is important to note that the `switchQDModel` switch in `paraCfgCurrent.txt` file can be set as `nistMeasurements` for NIST measurement based examples or `tgayMeasurements` for 11ay channel document [2] based examples. The `switchQDModel`[5] switch is used to toggle between two different Q-D models that are used to obtain the reflection loss and the diffuse or intra-cluster components for each specular ray (more details can be found in Section 6.1). At present, this software supports *lecture room*, *data center*, *parking lot*, *denser* and *spatial sharing* scenarios with `switchQDModel=nistMeasurements`, where reflection loss and diffuse components for each specular ray are obtained using NIST measurement based Q-D parameters. These Q-D parameters can be found in *.csv files placed in `/src/material_libraries/`. In addition to these scenarios, this software also generates mm-wave channels for *street canyon*, *living room*, *large hotel lobby*, and *open area hotspot* scenarios with `switchQDModel=tgayMeasurements`, where reflection loss and diffuse components for each specular ray are obtained using material properties and intra-cluster parameters, respectively, given in the 11ay channel document [2]. For reading convenience, the intra-cluster parameters (c.f. `/src/material_libraries/intraClusterTgayParameters.txt`) used for *street canyon*, *living room*, *large hotel lobby*, and *open area hotspot* scenarios are shown in Table 2. Each of the existing scenarios in `/src/examples/` can be launched by setting `scenarioNameStr` in `main.m`. For example, by setting `/examples/StreetCanyon`, the Q-D software generates mm-wave channel for street canyon. Further note that each example in `/src/examples/` returns the output in JSON format, which can be easily changed to `txt` format by considering `outputFormat` as `txt` in `paraCfgCurrent.txt` file. For better understanding, these scenarios and their predefined setting in `paraCfgCurrent.txt`, are also described in the following subsections.

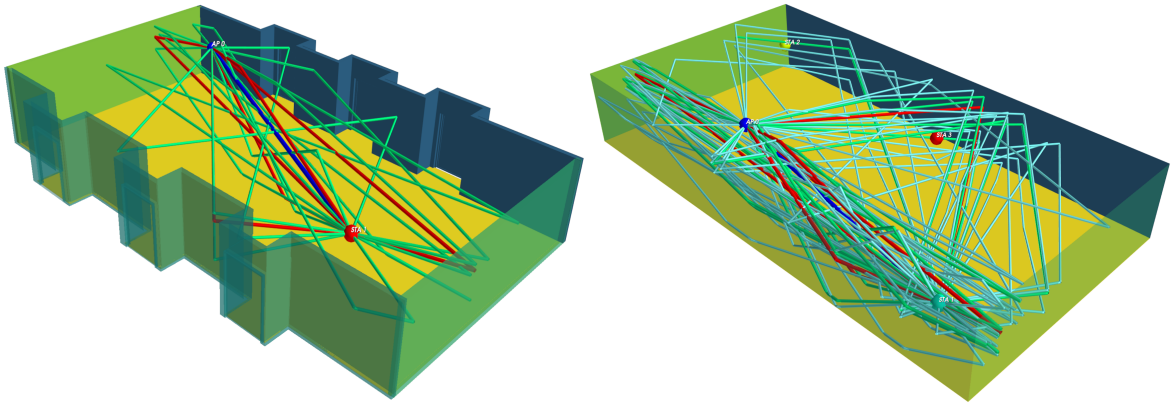
[5] If this parameter is missing in `paraCfgCurrent.txt` file, the software considers `nistMeasurements` as a default Q-D model.

Table 1: Examples available in the NIST Q-D Channel Realization Software and their default parameters.

Example	Scenario	Number of Nodes	Mobility	Rotation	Antenna Number Per Device	CAD File	Reflection Order	Measurements NIST/TGay/NA	Frequency (in GHz)	Diffuse Components	Material Library
LectureRoom	Indoor	2	No	No	1	LectureRoom.xml	1	NIST	60	Yes	materialLibraryLectureRoom.csv
BoxLectureRoom	Indoor	4	Yes	No	1	Box.xml	3	NIST	60	Yes	materialLibraryLectureRoom.csv
DenserScenario	Indoor	11	No	No	1	Box.xml	2	NIST	60	Yes	materialLibraryLectureRoom.csv
SpatialSharing	Indoor	4	Yes	No	1	Box.xml	2	NIST	60	Yes	materialLibraryLectureRoom.csv
L-Room	Indoor	2	Yes	No	1	L-Room.xml	2	NA	60	No	materialLibraryEmpty.csv
L-Room-rotation	Indoor	2	Yes	Yes	1	L-Room.xml	2	NA	60	No	materialLibraryEmpty.csv
L-Room-rotation-multiPAAs	Indoor	2	Yes	Yes	2	L-Room.xml	2	NA	60	No	materialLibraryEmpty.csv
DataCenter	Indoor	2	No	No	1	DataCenter.xml	2	NIST	60	Yes	materialLibraryDataCenter.csv
ParkingLot	Outdoor	2	Yes	No	1	ParkingLot.xml	1	NIST	28	Yes	materialLibraryParkingLot.csv
SimplifiedParkingLot	Outdoor	2	Yes	No	1	SimplifiedParkingLot.xml	2	NIST	28	Yes	materialLibraryParkingLot.csv
LivingRoom	Indoor	2	No	No	1	LivingRoom.xml	2	TGay	60	Yes	materialLibraryLivingRoom.csv
StreetCanyon	Outdoor	3	Yes	No	1	StreetCanyon.xml	2	TGay	60	Yes	materialLibraryStreetCanyon.csv
HotelLobby	Indoor	6	No	No	1	HotelLobby.xml	2	TGay	60	Yes	materialLibraryHotelLobby.csv
OpenAreaHotspot	Outdoor	16	No	No	1	OpenAreaHotspot.xml	2	TGay	60	Yes	materialLibraryOpenAreaHotspot.csv
CityBlock	Outdoor	2	Yes	No	1	CityBlock.xml	2	NA	60	No	materialLibraryCityBlock.csv

Table 2: Intra-cluster parameters as given in [2], where n, K, γ, λ, and σ represent the number of rays, K-factor, power decay time, arrival rate, and RMS for angle spread, respectively.

	Pre Cursor				Post Cursor				
Scenario	n	K(dB)	γ(ns)	λ(1/ns)	n	K(dB)	γ(ns)	λ(1/ns)	σ
OpenAreaHotspot	0	0	0	0	4	4	4.5e-9	3.1e8	5
StreetCanyon	0	0	0	0	4	4	4.5e-9	3.1e8	5
HotelLobby	0	0	0	0	6	10	4.5e-9	3.1e8	5
LivingRoom	6	11.5	1.25e-9	2.9e8	8	10.9	8.7e-9	1e9	10



(a) Lecture-room scenario (b) Simplified lecture-room scenario

Figure 17: Lecture-room scenarios.

5.1 Lecture Room

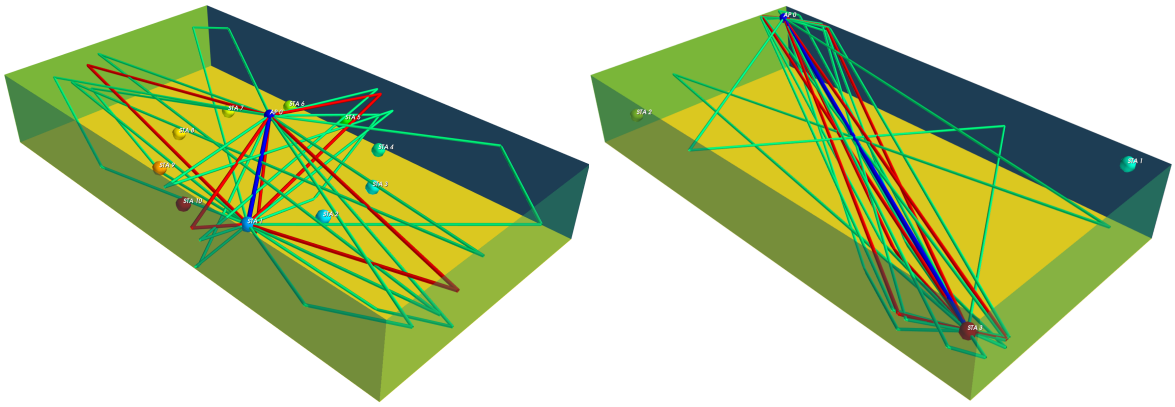
This example folder considers a complex architecture of lecture room defined by the `LectureRoom.xml` file. In this scenario, we consider the presence of two nodes, one is at a height of 2.5 m and other one is at a height 1.6 m. Both are assumed to be stationary inside the room, as shown in Figure 17a. Since this is an indoor scenario and all the nodes are present inside the room, the switch `indoorSwitch` is activated by assigning a value of '1' in `paraCfgCurrent.txt` file.

Further note that diffuse or intra-cluster components for each of the specular rays can be turned on or off by assigning `switchDiffuseComponent` = 1 or 0, respectively. When the switch `switchDiffuseComponent` = 1, the Q-D parameters given in `materialLibraryLectureRoom.csv` file (c.f. `/material_libraries/`) are used to generate diffuse components. These Q-D parameters given in the material library file are obtained using the measurements collected from state-of-the-art 60 GHz channel sounder available at NIST. Thus, the switch `switchQDModel` is considered as `nistMeasurements` in `paraCfgCurrent.txt` file.

5.2 Simplified Lecture Room

In contrast to the complex ***LectureRoom*** environment considered in the previous example, this ***BoxLectureRoom*** example considers a simple box scenario, i.e. an empty rectangular lecture room of size $19\times10\times3\,\mathrm{m}^3$ as shown in Figure 17b. This simple box scenario which consist of only six faces, enables us to consider multiple nodes along with their mobility as well as higher order of reflection while generating the channel between nodes due to very less computational time complexity. For demonstration purposes, in this example we consider four mobile nodes inside the room and generate rays up to third order of reflections.

Further note that the rectangular room is defined by the `box.xml` file given in



(a) Denser scenario (b) Spatial-sharing scenario

Figure 18: Lecture-room with denser spatial-sharing scenarios.

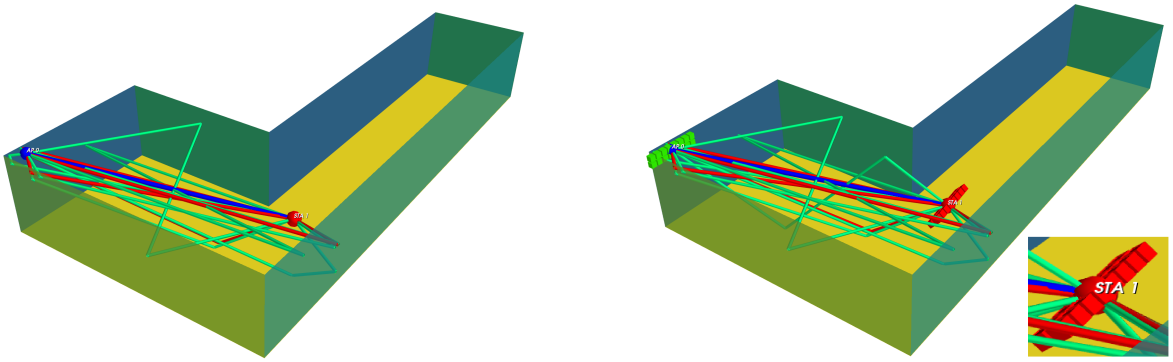
`/BoxLectureRoom/Input/`. This example considers four dynamic nodes and their positions are included as an input through `NodePosition`X`.dat`, $X = 1, 2, 3, 4$, files in `/BoxLectureRoom/Input/`, where one node is assumed to be at a height of 2.6 m and other three nodes are at 1.6 m. Moreover, user can also observe that `diffusePathGainThreshold` is set as -25 dB in the `paraCfgCurrent.txt` file, which means the software considers only diffuse components up to 25 dB below the specular component for each of the clusters.

5.3 Denser Scenario

As shown in Figure 18a, the **denserScenario** is also an indoor scenario where nodes are present inside the rectangular room of size $19\times10\times3\,\mathrm{m}^3$. However, in contrast to ***LectureRoom*** and ***BoxLectureRoom***, this example folder considers denser deployment of nodes where 11 static nodes are assumed inside the room. The user can also observe that apart from `NodePosition` files, the rest of the setting in `/DenserScenario/Input/` is similar to the one considered in the ***LectureRoom*** and ***BoxLectureRoom*** scenarios. More information about this scenario can be found in [11].

5.4 Spatial Sharing

The **spatialSharing** scenario was designed to study spatial sharing performance in an IEEE 802.11 ay use-case. In this example, two node pairs are considered inside a rectangular room of size $19\times10\times3\,\mathrm{m}^3$ in which each node pair comprises of one AP and one STA, as shown in Figure 18b. Both APs communicate with their associated STA using the same time and frequency resources. However, one STA is assumed to be dynamic with 100 different locations, as can be seen in `NodePosition3.dat` file in `/SpatialSharing/Input/` folder. Thus, `numberOfTimeDivisions` is considered as 100



(a) L-Room block scenario (b) L-Room with rotation scenario

Figure 19: L-Room scenarios under various configurations.

in `paraCfgCurrent.txt` file. Further, note that the value for `totalTimeDuration` is set as 100, which means the channel is simulated with a sampling time of 1 s. Further details about this scenario can be found in [11].

5.5 L-shaped room

This scenario was designed to study the blockage in an IEEE 802.11 ay use-case. As shown in Figure 19a, the **L-room** example considers an L-shaped room scenario, represented by `L-room.xml` file. In this example, the AP is attached to ceiling at a height of 3 m and a STA is considered to be dynamic with 200 different locations as defined in `NodePosition1.dat` file. Moreover, due to unavailability of Q-D parameters for this scenario, `switchDiffuseComponent` is set as 0, which means the software generates only specular rays. Users can also observe that `materialLibraryPath` is missing in `paraCfgCurrent.txt` file and for missing `materialLibraryPath`, the software considers `reflectionLoss` = 15 dB for each specular reflection. Note that `reflectionLoss` = 15 dB is defined in the `paraCfgCurrent.txt` file, which can be easily modified. In case this `reflectionLoss` parameter is missing, the software considers 10 dB (default) reflection loss for each specular reflection. More details about this scenario can also be found in [11].

5.6 L-shaped room with rotating nodes

The ***L-Room-rotation*** example folder is an extension of the previous ***L-Room*** scenario that includes the rotation feature. The position and the rotation of the dynamic STA are included as input of the software using `NodePosition1.dat` and `NodeRotation1.dat` files. The user can also observe that the position and the rotation of the AP in `NodePosition0.dat` and `NodeRotation0.dat` are fixed as [0.5,0.5,3] and [0,0,0], respectively (Figure 19b depicts in red the STA rotated PAA). It is also

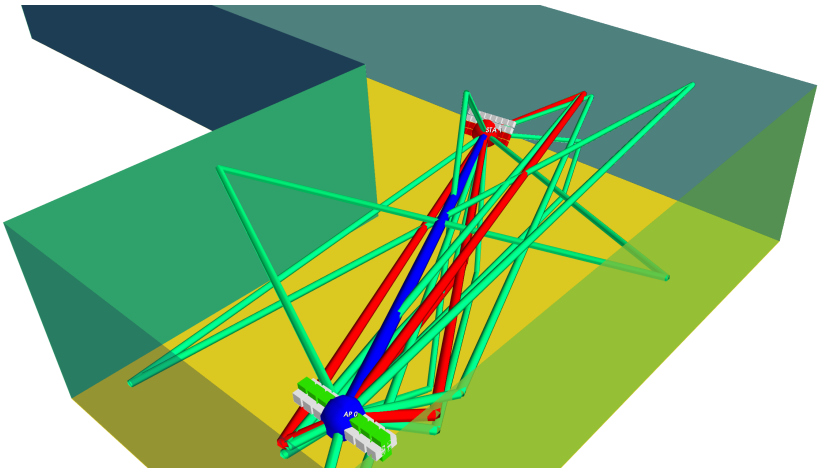


Figure 20: L-Room scenario with multiple PAAs and rotation

important to note that the software considers 10 dB (default) reflection loss for each specular reflection since `materialLibraryPath` and `reflectionLoss` parameters are missing in `paraCfgCurrent.txt` file. Apart from this, rest of the setting is same as considered in previous ***L-Room*** scenario.

5.7 L-shaped room with multi-PAA nodes

The input parameters in***L-Room-rotation-multiPAAs*** are identical to the ones considered in the previous ***L-Room-rotation*** scenario. However, in contrast to ***L-Room*** and ***L-Room-rotation*** example folders, each node in ***L-Room-rotation-multiPAAs*** is equipped with two PAAs, as shown in Figure 20. Node 0, the AP, has two PAAs centered in the same position but oriented in a different direction as defined in `NodePaa0.dat`. Node 1, the STA, has two PAAs distant 0.25 m along the z-axis and they are oriented in opposite directions as defined in as defined in `NodePaa1.dat`.

5.8 Data Center

As shown in Figure 21, the ***DataCenter*** scenario is created based on [12], where the room dimension is 18 m×18 m×2.56 m and rack heights are in between 1.96 m and 2 m. It is important to note that this indoor scenario is one of the 802.11 ay use cases where the Tx and the Rx are assumed to be static and are placed on the rack-tops to have inter-rack connectivity if the Ethernet wired connection disconnected or broken. Thus, in this example folder, we consider the placement of both Tx and Rx on the rack-tops at [8.39, 15.4, 2.19] and [7.92, 4.4, 2.19], respectively. Further, note that the diffuse parameters considered in material library file `materialLibraryDataCenter.csv` are obtained using the channel measurements [12] carried out by NIST. Therefore, the switch `switchQDModel` is considered as `nistMeasurements` in `paraCfgCurrent.txt`

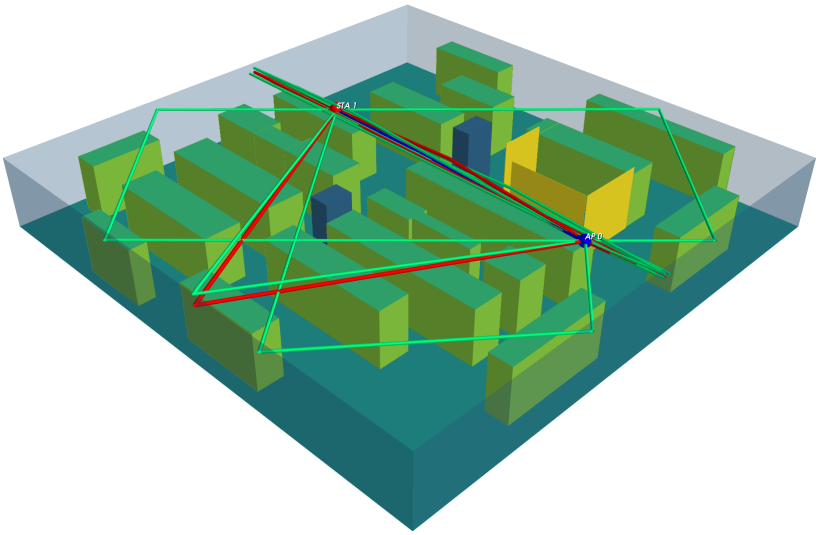


Figure 21: Data center scenario where both nodes are placed on rack-tops.

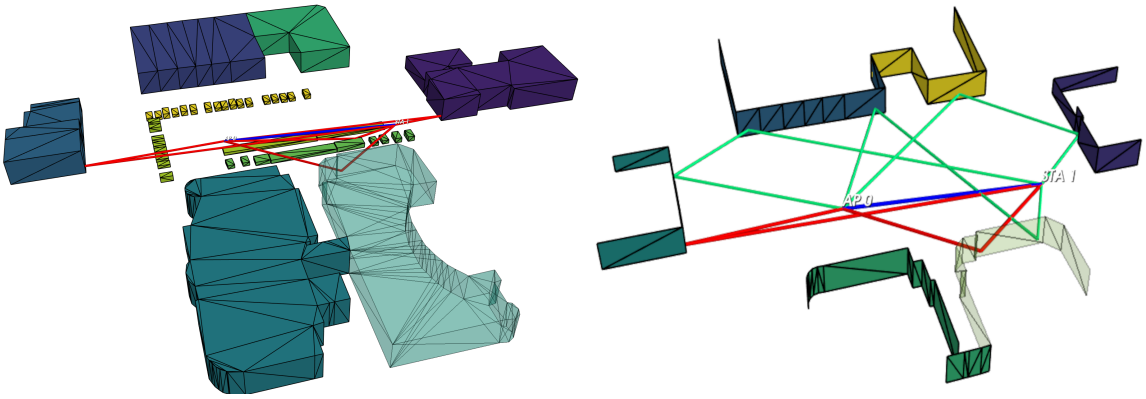
file.

5.9 Parking Lot

The ***ParkingLot*** is an outdoor urban scenario shown in Figure 22a where the parking lot is surrounded by the buildings. In this example, Open Street Map (OSM) with coordinates (latitude and longitude angles) is used to generate the CAD model of the location, which is considered for measurement campaign in downtown Boulder, Colorado [13]. In this scenario, the Tx is fixed at position $[0, 0, 2.5]$ and the Rx is considered to be dynamic with 61 different locations as defined in `NodePosition1.dat` file. Moreover, in `paraCfgCurrent.txt` file, `indoorSwitch` is set as 0 since this an outdoor scenario and `totalNumberOfReflections` is set as 1 to reduce the software execution time since this is very complex scenario with large number of surfaces. Further note that the diffuse parameters considered in material library file `materialLibraryParkingLot.csv` are obtained using the measurements collected with our 28 GHz directional channel sounder. Therefore, the switch `switchQDModel` is set as `nistMeasurements`.

5.10 Simplified Parking Lot

The above ***ParkingLot*** example considers a complex environment with large number of surfaces, as shown in Figure 22a. Thus, to reduce the number of triangles, the ***SimplifiedParkingLot*** example considers a simplified CAD model as shown in Figure 22b for ray tracing, where blocks corresponding to the cars and all the surfaces which are not directly exposed to the parking lot, are discarded. This simplified CAD model



(a) Parking-lot scenario (b) Simplified parking-lot scenario

Figure 22: Parking-lot scenarios

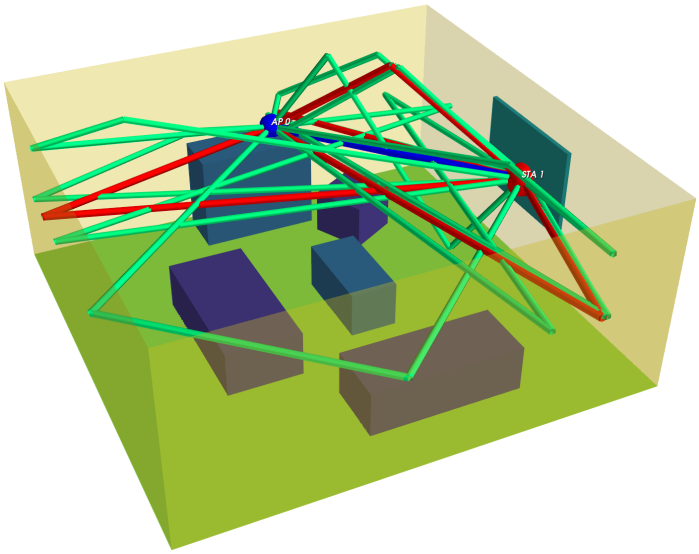


Figure 23: Living-room scenario.

enables us to consider higher order of reflection while generating the channel between Tx and Rx. Therefore, in contrast to previous example, `totalNumberOfReflections` is considered as 2 in `paraCfgCurrent.txt` file. Moreover, the rest of the parameters in `paraCfgCurrent.txt` file is same as considered in previous example.

5.11 Living Room

As shown in Figure 23, the indoor ***LivingRoom*** scenario considers a square living room of size $7 \times 7 \times 3\,\mathrm{m}^3$, where a static Tx is communicating with a static Rx. The Tx is placed on the table at a height of 1.6 m, whereas the Rx is attached to television at a

Table 3: Material library for living room environment.

Reflector	Material	Relative Permittivity
Ceiling	Plaster board	$6.25 + 0.3j$
Wall	Concrete	$4 + 0.2j$
Floor	Concrete	$4 + 0.2j$
Sofa	Wood	$1.5761 - 0.0962j$
TV	Glass	$5.2839 - 0.2538j$
Table	Wood	$1.5761 - 0.0962j$

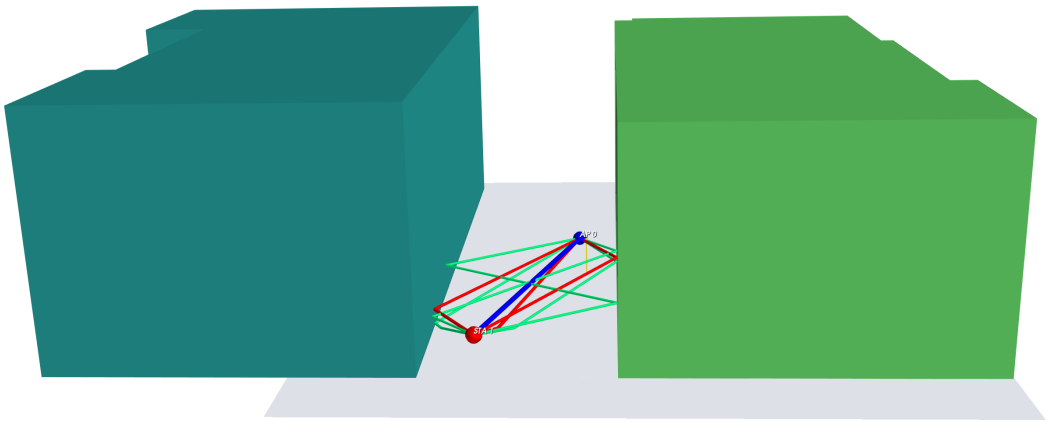
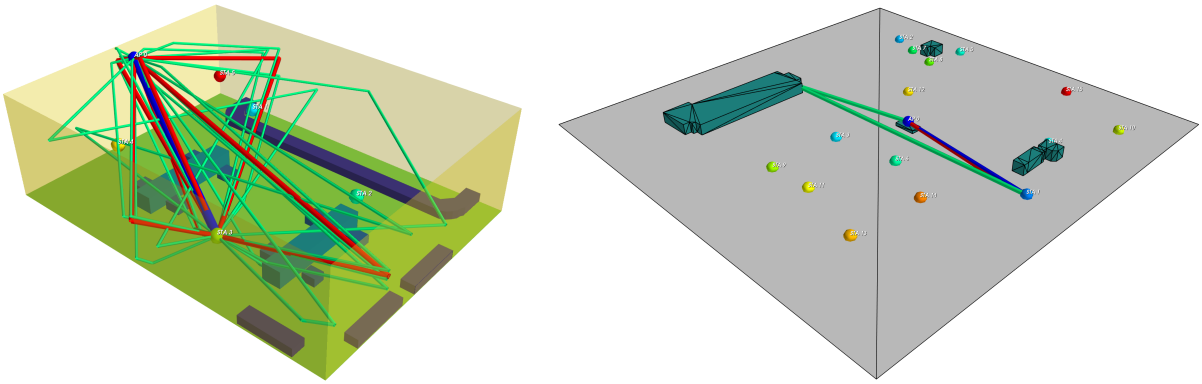


Figure 24: Street-canyon scenario.

height of 1.5 m for receiving uncompressed high-definition video. It is worth noting that the setting in `paraCfgCurrent.txt` file is same as ***BoxLectureRoom*** scenario except `switchQDModel` and `materialLibraryPath`. Since this scenario is based on 802.11 ay channel document [2], the switch `switchQDModel` is set as `tgayMeasurements`, which obtains reflection loss and diffuse components for each specular ray using material properties and intra-cluster parameters given in `materialLibraryLivingRoom.csv` and `intraClusterTgayParameters.txt`, respectively. For reading convenience, the material library used in `materialLibraryLivingRoom.csv` is also given in Table 3.

5.12 Street Canyon

As shown in Figure 24, the ***StreetCanyon*** is an urban street canyon scenario, one of the 802.11 ay use cases, where one AP, which is attached to a lamppost at the height of 6 m, is serving two dynamic STAs each at height of 1.5 m. Note that in this scenario, one STA is walking on the sidewalk and the other one is traveling on the road. Their positions are provided as an input through `NodePosition1.dat` and `NodePosition2.dat` files in `/StreetCanyon/Input/` folder. Similar to ***LivingRoom***, `switchQDModel` in `paraCfgCurrent.txt` file is set as `tgayMeasurements`. Moreover,



(a) Hotel lobby scenario.

(b) Open area hostpot scenario.

Figure 25: Hotel Lobby and Open Area hotspot scenarios

`materialLibraryPath` is set as `material_libraries/materialLibraryStreetCanyon.csv`, which includes building and ground materials as concrete and asphalt, respectively.

5.13 Hotel Lobby

As shown in Figure 25a, the ***HotelLobby*** example considers a building structure of large hotel lobby of size 20 m×15 m×6 m, where the AP is attached close to the ceiling at height of 5.5 m. This scenario represents typical indoor scenario considering large hall with multiple users [2]. Therefore, for demonstration purposes, the presence of five active downlink STAs is considered inside the hall. Moreover, these STAs are assumed to be static and deployed randomly within the hall. Their positions can be found in the `NodePosition1.dat`, `NodePosition2.dat`, `NodePosition3.dat`, `NodePosition4.dat` and `NodePosition5.dat` files. Similar to ***LivingRoom*** and ***StreetCanyon*** scenarios, `switchQDModel` in `paraCfgCurrent.txt` file is set as `tgayMeasurements`. Moreover, `materialLibraryHotelLobby.csv` file is used as a material library that considers ceiling, wall, floor, sofa, chair and table materials as plaster board, concrete, concrete, wood, wood, and wood, respectively.

5.14 Open Area Hotspot

As shown in Figure 25b, the ***OpenAreaHotspot*** example considers an outdoor open large area with very few buildings which are located very far from each other. The Tx that acts as an AP, is fixed at a height of 6 m on the roof of a building located in the middle of the scenario. Moreover, 15 STAs are deployed randomly in the scenario for downlink reception. These STAs, each at height of 1.5 m, are assumed to be static and their positions can be found in the `NodePosition` files given in `/OpenAreaHotspot/Input/` folder. Similar to ***LivingRoom***, ***StreetCanyon*** and ***OpenAreaHotspot***, `switchQDModel` in `paraCfgCurrent.txt` file is set as `tgay-`

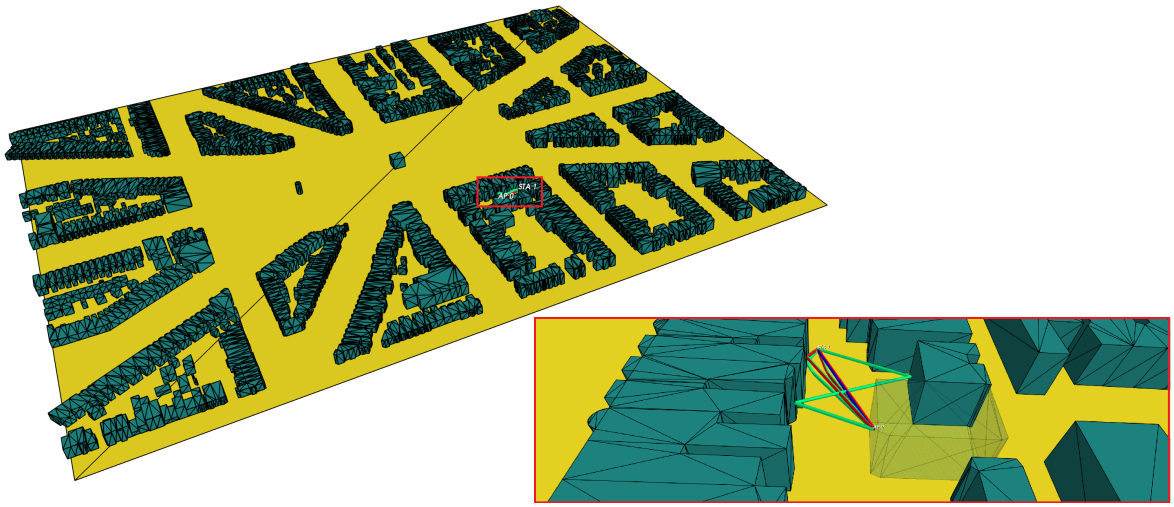


Figure 26: City block scenario where one node is attached to a building and other node is mobile on the ground.

`Measurements`. Moreover, `materialLibraryOpenAreaHotspot.csv` file, which includes building and ground materials as concrete and asphalt, respectively, is used to calculate the reflection loss for each specular ray.

5.15 City Block

In addition to above scenarios, this software also introduces the ***CityBlock*** example, which considers an outdoor location where nodes are present outside the buildings of a city block. The city block is defined by the `CityBlock.xml` file, which is generated using OSM. Since this is an outdoor scenario, the switch `indoorSwitch` is deactivated using the value 0. As shown in Figure 26, this is a very large scenario with over hundred thousand triangles. Not all triangles are useful for computation, as some triangles are very far from the nodes and even if we have specular ray reflection, their contribution would marginal due to the mm-wave properties. Thus, these triangles are eliminated based on the distance from the `referrencePoint`, which can be considered close to the nodes. The `referrencePoint` represents the center of the limiting sphere and in this example, [-57.7,97,3] is set as the x, y, z coordinates of the center, which is close to the nodes. Moreover, `selectPlanesByDist` is considered as 20 m, which defines the radius of the limiting sphere.

Further note that the switch `switchDiffuseComponent` in `paraCfgCurrent.txt` file is set as 0 since we do not have intra-cluster parameters for this scenario. Apart from this, the switch `switchQDModel` is set as `tgayMeasurements` since we calculate the reflection loss using the material properties given in material library (c.f. `materialLibraryCityBlock.csv`).

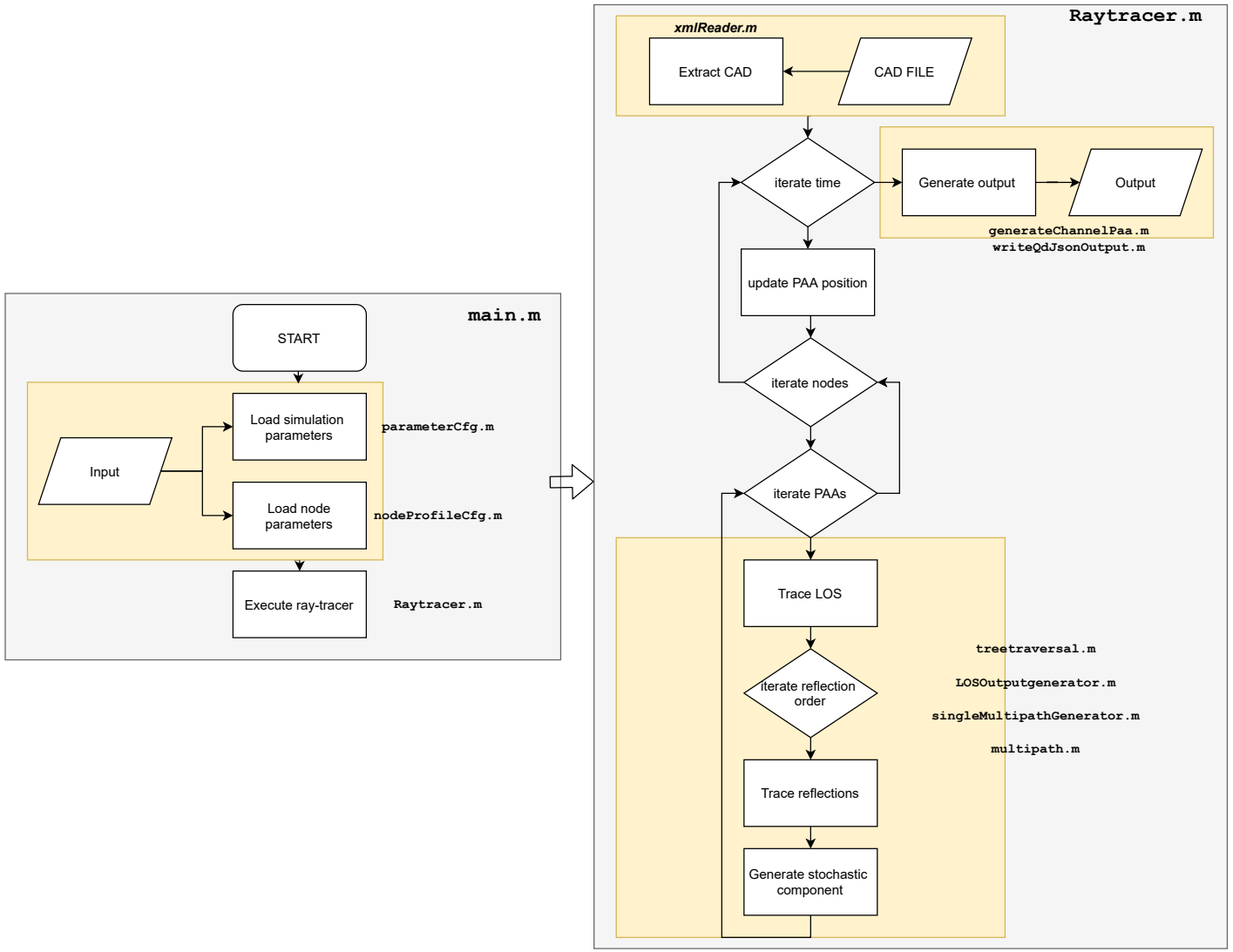


Figure 27: Flowchart of Raytracer.m.

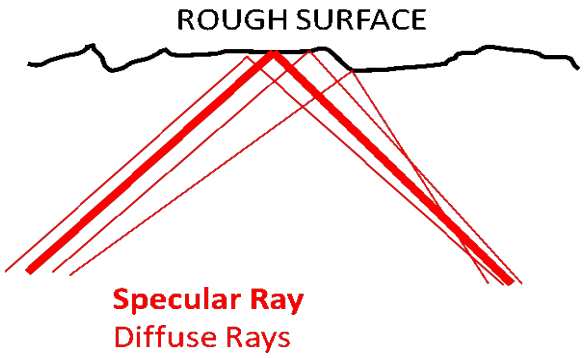


Figure 28: Rough surface scattering of deterministic ray (thicker red line).

6 Implementation

This section describes the detail of each implementation block used in the NIST Q-D Channel Realization Software. An overview of the code and the main functions are illustrated in Figure 27.

6.1 QD Model

This software supports two Q-D models to provide a wide range of scenarios and applications to simulate. One is based on statistical distributions using NIST measurements and other one is based on 802.11 ay channel document [2].

The NIST Q-D Channel Realization Software implements the specular rays using the method of images in both NIST and TGay Q-D models. The specular rays are the strongest MPCs between the Tx and the Rx. In the DDIR, there are other weaker rays, which are generated due to the scattering of the specular rays from objects or surfaces. These rays are called *diffuse* rays or *intra-cluster* components since they form clusters around the reflected rays. Every cluster is composed of a specular component and multiple diffuse components, as shown in Figure 28.

In both NIST and TGay models, a total of L diffuse MPCs are generated for each of the specular rays. The diffuse components are further divided into L_{pre} precursor and L_{post} postcursor components. The precursor components arrive before the specular component and the postcursor components arrive after the specular component. Thus, each cluster has one specular component, L_{pre} precursor and L_{post} postcursor components. These clusters are characterized by the following parameters:

- **K-factor** : The relative strength of the deterministic (or specular) component with respect to precursor or post-cursor diffuse components is gauged through

the K-factor as [12],

$$K\text{-factor} = \frac{PG_{\text{det}}}{\sum_{i=1}^{L_{\text{pre,post}}} PG_i}, \tag{7}$$

where PG_{det} and PG_i denote the path gains of the deterministic (or specular) component and ith precursor or postcursor diffuse component, respectively. Note that the K-factor gives the notion of how much power is scattered from the deterministic component.

- **Rate of Decay** : The power levels of diffuse components exponential decrease from the deterministic component and the rate of decay is represented by γ.
- **Delay Spread** : The delay spread is modeled using the Poisson distribution with mean and variance λ.
- **Angular Spread** : The angular spread in both elevation and azimuth planes follows a Laplacian distribution with zero mean and variance Θ^2.

Despite the conceptual similarities, the two models exhibit a different behaviour when generating the reflection loss of the specular components. Figure 30 gives an overview of the two model behaviour, which will be described in detail in the following of this section.

6.1.1 NIST Q-D model

The NIST Q-D model is developed using the channel measurements conducted in several environment such as a lecture room, a parking lot, and a data center environments.

Diffuse Components

The diffuse components are generated when `switchDiffuseCompenent=1`. The flow chart of NIST Q-D model is given in Figure 29. Based on some experimental evidence, we suggest to use $L_{\text{pre}} = 3$ and $L_{\text{post}} = 16$, although these numbers may vary in different locations and models, hence a total of 19 diffuse MPCs are generated for each specular rays.

The K-factor, γ and λ, are randomly generated using the Rician distribution which is characterized by two parameters s and σ. Moreover, the angular spread in both elevation and azimuth planes follows a Laplacian distribution with zero mean and variance Θ^2, where Θ is randomly generated using the Rician distribution. The detailed description of NIST Q-D model can also be found in [3].

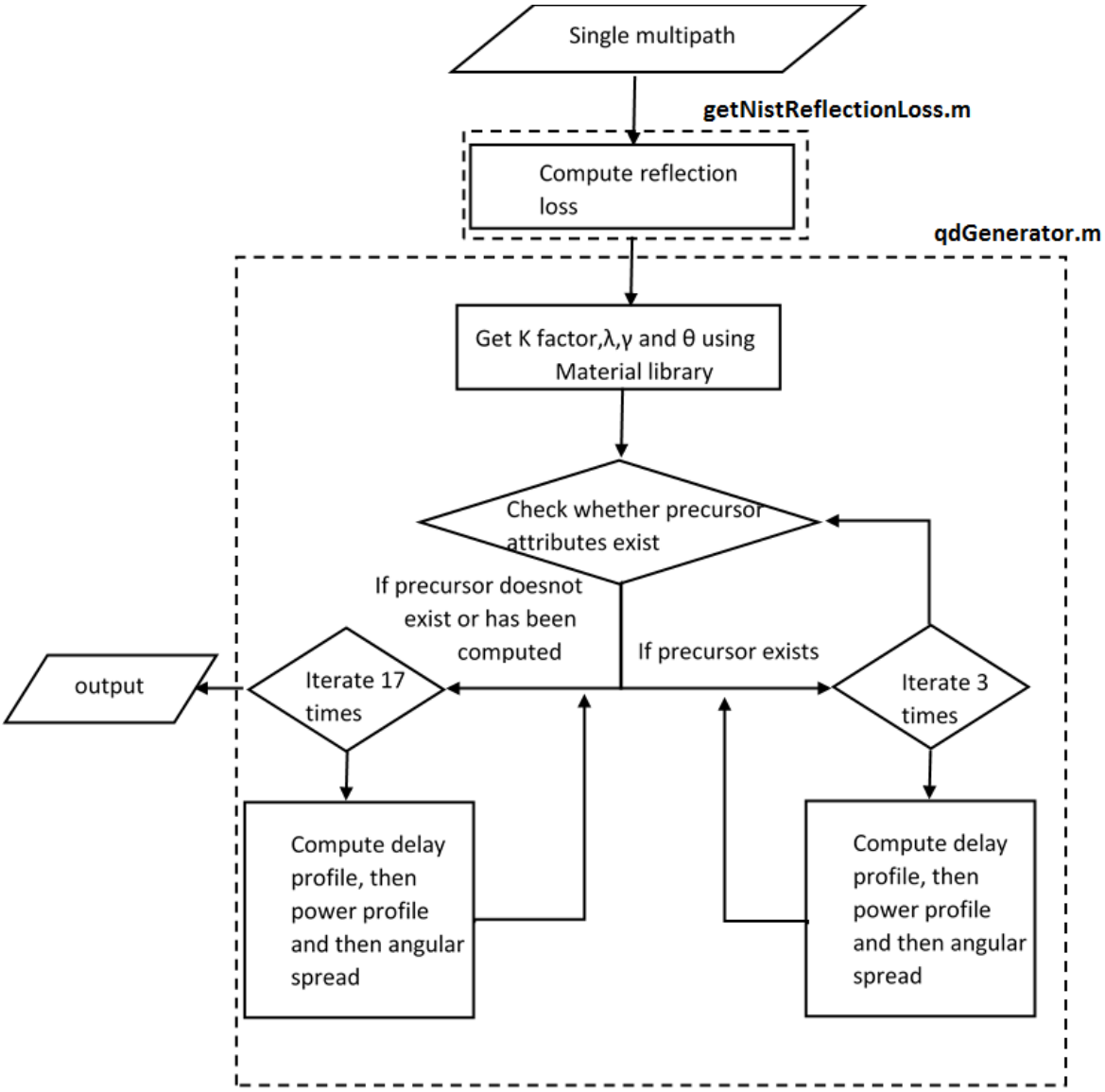


Figure 29: Flow chart of NIST Q-D model.

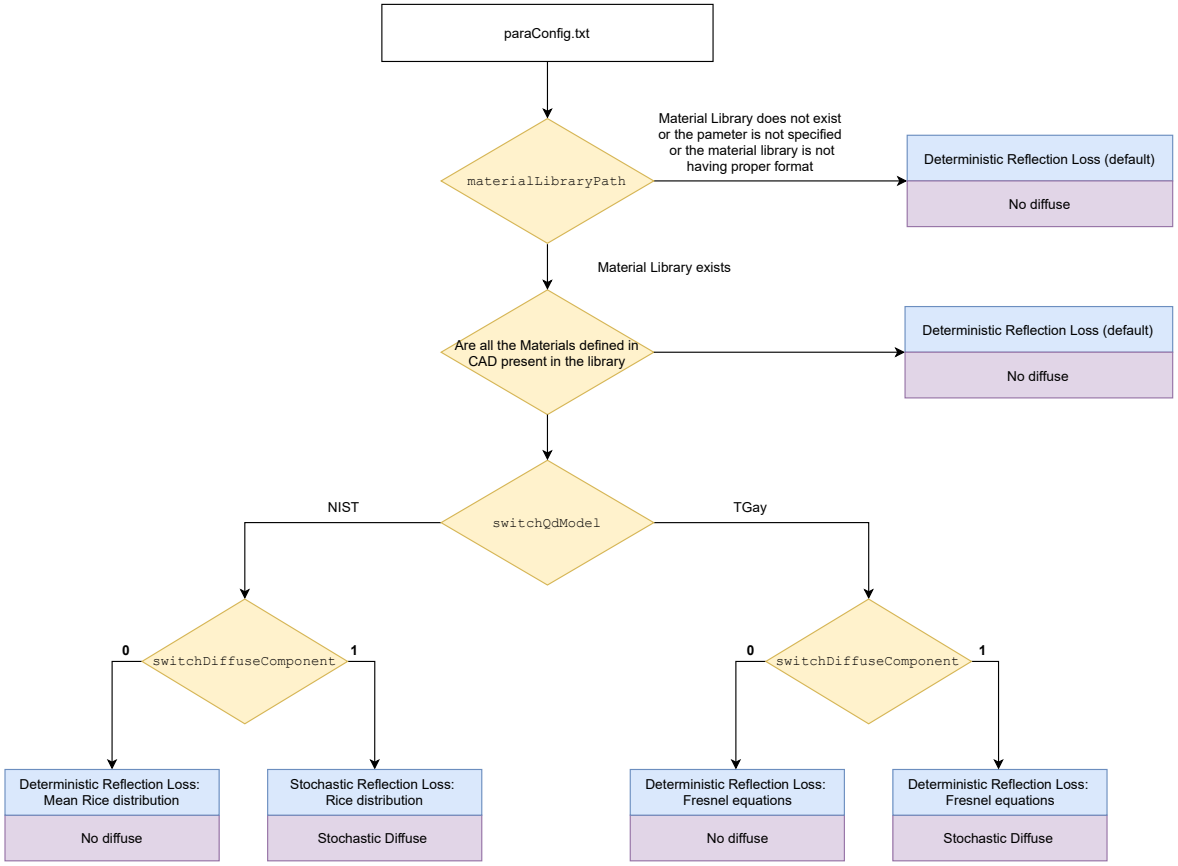


Figure 30: Flowchart of the different Q-D models implemented in the NIST Q-D Channel Realization Software.

Specular Components

The reflection loss is randomly generated using a Rician distribution, for each reflector present in the environment. When the NIST Q-D Channel Realization Software is used without diffuse component (`switchDiffuseCompenent=0`), the reflection loss is set to the mean of the Rician distribution to provide a deterministic channel realization. The Rician (s, σ) parameters for each of the reflectors in lecture room, parking lot and data center are provided in material library files given in `/src/material_libraries/`.

Material Library

The material library stores the parameters to generate both specular and diffuse components. If the material library path is not defined or any one of the materials defined in the CAD is not found in the material library or the material library file format is not correct, the material library is disabled. For these reasons, the diffuse components won't be generated. In this case the software is set to operate only generating specular components with deterministic reflection loss, which can be configured thank to the parameter `reflectionLoss`.

For reading convenience, the material library file attributes are described below:

- `Reflector` – denote the name of the reflector present in the environment.
- `n_Precursor` and `n_Postcursor` – denote the number of precursor and postcursor components.
- `s_K_Precursor` and `sigma_K_Precursor` – denote noncentrality and scale parameters of Rician distribution, which are used to obtain K-factor for precursor components.
- `s_K_Postcursor` and `sigma_K_Postcursor` – denote noncentrality and scale parameters of Rician distribution, which are used to obtain K-factor for postcursor components.
- `s_gamma_Precursor` and `sigma_gamma_Precursor` – represent noncentrality and scale parameters of Rician distribution, which are used to obtain γ for precursor components.
- `s_gamma_Postcursor` and `sigma_gamma_Postcursor` – represent noncentrality and scale parameters of Rician distribution, which are used to obtain γ for postcursor components.
- `s_sigmaS_Precursor` and `sigma_sigmaS_Precursor` – denote noncentrality and scale parameters to obtain Rician-distributed σ_s for precursor components, which is subsequently used to calculate the power-delay decay standard deviation using Normal distribution with mean zero and variance σ_s^2.

- `s_sigmaS_Postcursor` and `sigma_sigmaS_Postcursor` – denote noncentrality and scale parameters to obtain Rician-distributed σ_s for postcursor components, which is subsequently used to calculate the power-delay decay standard deviation using Normal distribution with mean zero and variance σ_s^2.
- `s_lambda_Precursor` and `sigma_lambda_Precursor` – represent noncentrality and scale parameters of Rician distribution, which are used to obtain λ for precursor components.
- `s_lambda_Postcursor` and `sigma_lambda_Postcursor` – represent noncentrality and scale parameters of Rician distribution, which are used to obtain λ for postcursor components.
- `s_sigmaAlphaAz` and `sigma_sigmaAlphaAz` - denote noncentrality and scale parameters of Rician distribution, which are used to obtain Θ for azimuth plane.
- `s_sigmaAlphaEl` and `sigma_sigmaAlphaEl` - denote noncentrality and scale parameters of Rician distribution, which are used to obtain Θ for elevation plane.
- `s_RL`, `sigma_RL`, and `mu_RL` - denote noncentrality parameter, scale parameter and mean of Rician distribution, which are used to obtain reflection loss. Note that material library included the mean `mu_RL` to speed up the code. However, it can be easily obtained by substituting `s_RL` and `sigma_RL` in the following expression

$$\texttt{mu\_RL} = \texttt{sigma\_RL}\sqrt{\frac{\pi}{2}}L_{1/2}\left(-\frac{(\texttt{s\_RL})^2}{2(\texttt{sigma\_RL})^2}\right), \tag{8}$$

  where $L_{1/2}(\cdot)$ is Laguerre polynomial.

6.1.2 TGay Q-D Model

The TGay Q-D model is developed following the TGay channel model document [2].

Diffuse Component

The 802.11 ay channel document [2] proposes tabulated values for K-factor, γ, λ, which are relative to the scenario. The AoA and the AoD spread for each intra-cluster component of specular ray are generated as independent Gaussian distributed random variables with mean μ and variance σ^2, where μ is equal to the corresponding angle of the specular ray.

Specular Component

The reflection loss for each specular ray is computed thanks to the Fresnel equation, depending on the incident angle(s), the polarization and the material properties. More

specifically, for each reflection, the NIST implementation first obtains the reflectances for vertical and horizontal polarization using the incident angle and relative permittivity of the material in the Fresnel equation. Subsequently, the reflection loss for each reflection is calculated by the taking the average of the vertical and horizontal reflectances.

Material Library

The K-factor, γ, λ, and σ parameter values used to generate intra-cluster precursor and postcursor components are given in the TGay channel document [2] for various indoor and outdoor scenarios. These values are included in `intraClusterTgayParameters.txt` file given in `/src/material_libraries/`. For better clarity, `intraClusterTgayParameters.txt` file attributes are described below.

- `Scenario` : defines name of the environment.
- `nPre` : denotes number of precursor components.
- `KfactorPre` : denotes K-factor for precursor components.
- `gammaPre` : denotes power decay time for precursor components.
- `lambdaPre` : denotes arrival rate for precursor components.
- `nPost` : denotes number of postcursor components.
- `KfactorPost` : denotes K-factor for postcursor components.
- `gammaPost` : denotes power decay time for postcursor components.
- `lambdaPost` : denotes arrival rate for postcursor components.
- `sigma` : denotes RMS value for angular spread.

As mentioned earlier, in TGay channel document based Q-D model, the calculation of reflection loss for each reflection considers the angle of incidence and the relative permittivity of the reflector material. The relative permittivity values of various reflector materials in street canyon, open area hotspot, living room, and hotel lobby scenarios are provided in material library files given in `/src/material_libraries/`. For better clarity, material library file attributes are also described below.

- `Reflector` : denotes the name of the reflector present in the environment.
- `Material` : denotes the material of the reflector.
- `RelativePermittivity` : denotes the relative permittivity of the material.

6.2 Optimized Ray Tracing for Multi-Point DDIR

NIST Q-D Channel Realization Software provides the details of the magnitude, phase, and time of arrival, DoD and DoA of individual propagation paths, which are key elements of a mm-wave MIMO channel. These parameters are provided for each Tx-Rx PAA combination since the NIST Q-D Channel Realization Software provides a scalable multi-PAAs Q-D channel model implementation based on the assumption that for PAAs separated by a distance smaller than the correlation distance d_{fc}, share the same DDIR but with phase difference. For this reason, the input provided in the file `NodePaa`X`.dat` is pre-processed before performing the ray tracing. A grouping algorithm finds clusters of PAAs, which experience the same deterministic component to optimize the run-time execution.

The grouping algorithm is described in Algorithm 1. During the initialization, the PAAs pairwise distance d is computed. Based on the distance d, the algorithm is divided in three main blocks:

1. First, the PAAs at a distance smaller or equal to d_{fc} are selected and the PAA surrounded by the highest number of PAAs is selected as the center of a cluster. This grouping continues until all the selected PAAs are assigned to a cluster.

2. In the second step the PAAs that have a distance between d_{fc} and d_c are selected. If any of these selected PAAs have been considered as the center of a cluster in the previous step, if possible new PAAs are added into the cluster. Otherwise new clusters are formed. The grouping continues until all the selected PAAs are assigned to a cluster.

3. All the PAAs that are not assigned to a cluster are managed independently.

While the algorithm works as described, the second step is in this deliverable disabled by setting $n_{\text{fc}} = n_c$.

6.3 CAD Data Extraction

6.3.1 CAD Format

CAD file consists of geometrical information of a given environment. These geometrical properties can be represented as point clouds, lines, and/or polygons. In the NIST Q-D Channel Realization Software, AMF file format is used as the standard-form of input CAD file format, which represent geometrical structures as triangles. In addition to geometrical properties, AMF file format can also store material properties of each triangle. The output of the CAD data extraction block is the set of triangles, which are described by the vertices in x, y, z format and the material properties. For example, a cube has 6 sides and it is represented as a set of 12 triangles, as shown in Figure 31. Once we extract the three vertices of a triangle, we can construct its plane equation.

Algorithm 1 PAA clustering

Require: `nodePosition`, `paaPosition`, n_c, n_{fc}

for each node n **do**

▷ Initialization

  Initialize array of PAAs: `paaList = [1,2, ..., NPAA]`

  Initialize array of PAAs clustered: `paaClustered = []`

  Initialize array of flags: `type` = []

  # `type` = 0: $d \leq \lambda/2 \cdot n_{\text{fc}}$,

  # `type` = 1: $\lambda/2 \cdot n_{\text{fc}} < d < \lambda/2 \cdot n_c$

  # `type` = 2: $d \geq \lambda/2 \cdot n_c$

  Compute pairwise distance d between PAAs

▷ Group PAAs with $d \leq d_{\text{fc}}$

  Find array of PAAs `paa2cluster` such that $d \leq \lambda/2 \cdot n_{\text{fc}}$

  while `paa2cluster` is not empty **do**

    Add PAA with most connections in array `centroid`

    Compute centroid position $\mathtt{p}_c$ from `nodePosition` and `paaPosition`

    Add PAAs connected with centroid into the PAA clustered array `paaClustered` and label them as `type =0`

    Remove PAAs connected with centroid from `paa2cluster`

  end while

▷ Group PAAs with $d_{\text{fc}} < d < d_c$

  Find array of PAAs `paa2cluster` such that $\lambda/2 \cdot n_{\text{fc}} < d \leq \lambda/2 \cdot n_c$

  if Any of the PAAs in `paa2cluster` is in `centroid` **then**

    Add PAAs connected with centroid into the PAA clustered array `paaClustered` and label them as `type =1`

    Remove PAAs connected with centroid from `paa2cluster`

  end if

  while `paa2cluster` is not empty **do**

    Add PAA with most connections in array `centroid`

    Compute centroid position from `nodePosition`, `paaPosition`

    Add PAAs connected with centroid into the PAA clustered array `paaClustered` and label them as `type =1`

    Remove PAAs connected with centroid from `paa2cluster`

  end while

▷ PAAs with $d \geq d_c$

  No cluster found for PAAs in `paaList = [1,2, ..., NPAA]` and not included in `paaClustered`

  Compute PAAs position and label them as `type =2`

end for

Return: $\mathtt{p}_c$, `type`

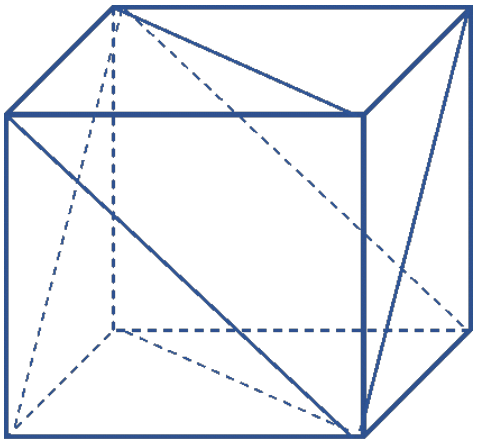


Figure 31: A cube is described by 12 triangles in the AMF file format.

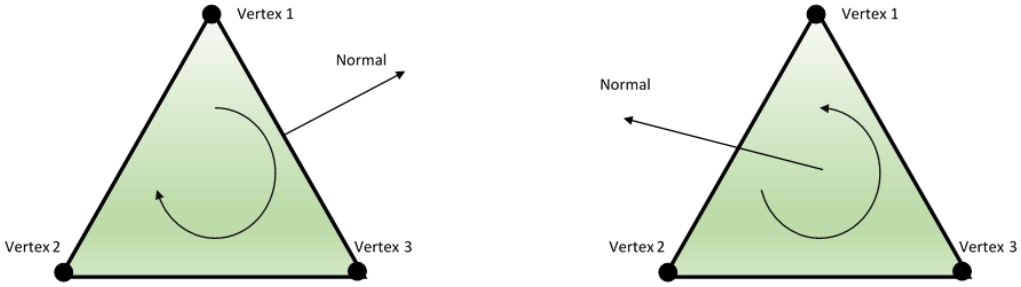


Figure 32: A triangle and its normal (a) when `IndoorSwitch` has a value of 1. The normal is taken in clockwise direction (b) when `IndoorSwitch` has a value of 0. The normal is taken in counterclockwise direction.

The normal of this plane equation is given by right hand rule according to AMF format guidelines. If the normal is to be taken in clockwise direction then `IndoorSwitch` in `paraCfg` (or `paraCfgCurrent`) file should have a value of '1'. For counterclockwise direction this value should be '0'. These are shown in Figures 32a and 32b.

6.3.2 Code Structure

The functions `RayTracer.m`, `XmlReader.m` and `Xml2Struct.m` are used to extract the CAD information. `Raytracer.m` is the function that calls `XmlReader.m` function as shown in the flow chart in Figure 33. `XmlReader.m` takes `IndoorSwitch`, `MaterialLibrary`, r and reference point as input parameters, where `IndoorSwitch` defines how the normal is oriented. Normal of a triangle either comes out of the plane (vertices defined in clockwise direction using right hand rule) or into the plane (vertices are in anti-clockwise direction using right hand rule). `MaterialLibrary` is a table which has the material properties. The quantity r and the reference point denote the

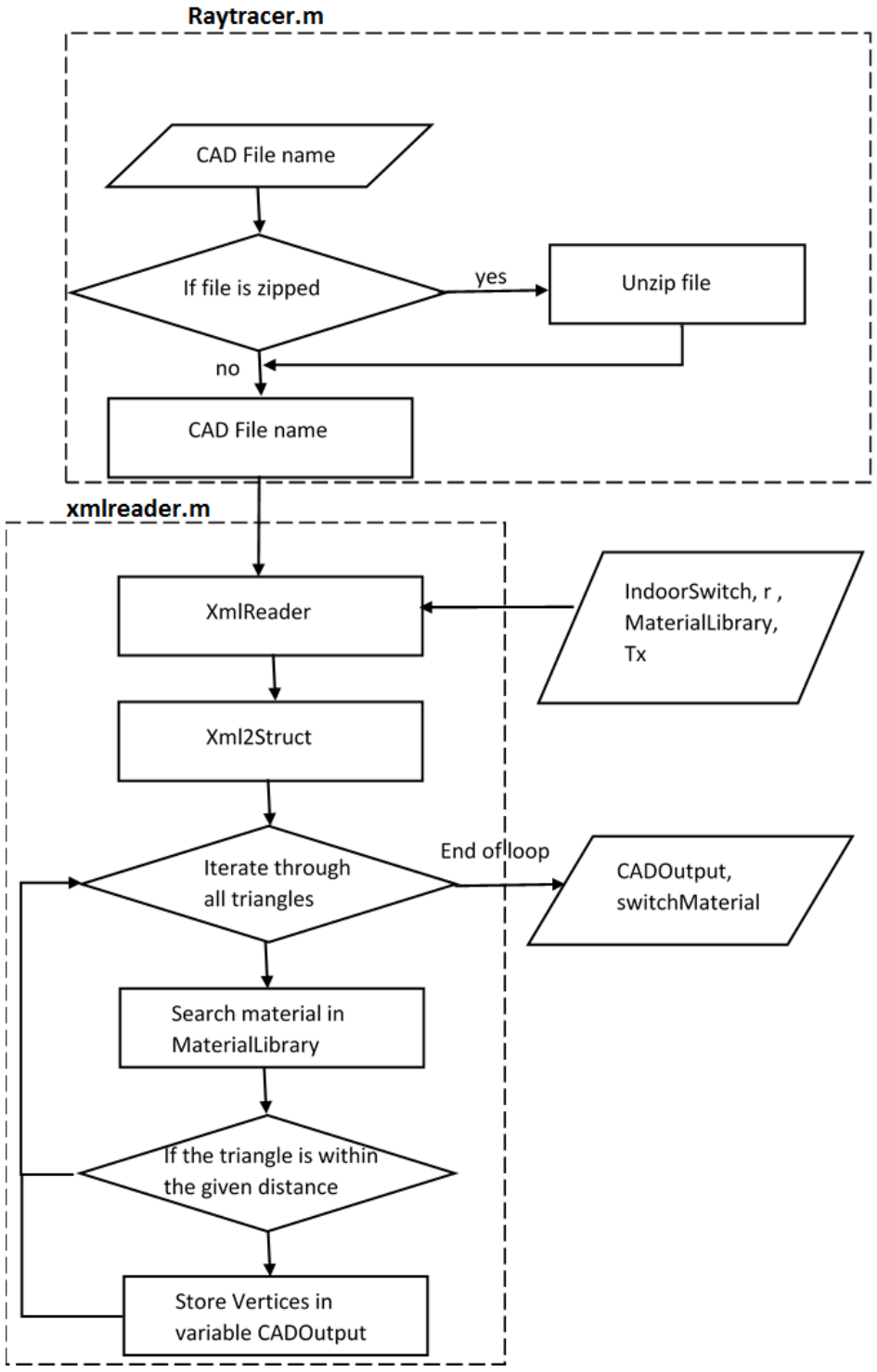


Figure 33: Flow chart of CAD extraction.

x1	y1	z1	x2	y2	z2	x3	y3	z3	a	b	c	d	Material Id

Figure 34: A row of `CADOutput` variable.

radius of limiting sphere and the center of the limiting sphere, respectively. These parameters are also described in section 6.3.3 in detail. The MATLAB `Xml2Struct.m` function extracts the XML data and converts it into a struct variable `s`. In the struct variable `s` the triangles and the corresponding vertices are stored in a chronological order. We iterate through all the triangles and search if the material is present in the material library. If it is present, then it is associated with the triangle. The information of the triangles is finally stored in `CADOutput` variable. The `CADOutput` variable is a Two-Dimensional (2D) array with 14 columns and the number of rows that is equal to the number of triangles. One row of `CADOutput` variable is shown in Figure 34.

In the first 9 columns, $\mathtt{x}_i$, $\mathtt{y}_i$, $\mathtt{z}_i$ are x, y, z coordinates of the i-th vertex of the triangle. A plane equation is given as $ax + by + cz + d = 0$. The variables a, b, c, d in the columns 10 to 13 are x, y, z coefficients and the constant, respectively. The variable `Material Id` in the last column helps in identifying the material associated with the triangle.

6.3.3 Omitting Triangles based on Distance from a Reference Point

The function `distanceLimitation.m` is used to omit the triangles, which are far from the defined reference point to avoid ray tracing of rays that do not contribute significantly in the received power. If we consider these triangles, it would increase significantly the computational time.

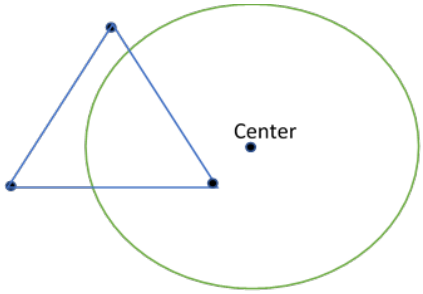
To identify the triangles contributing significantly in the propagation description we intersect the triangles with a sphere of radius r considering the reference point as the center of the sphere. The triangles that intersect the sphere are considered in the ray tracing, while the others are discarded. There are three possible cases in which a triangle intersects the sphere.

Case 1 : Either of the three vertices are within the sphere -

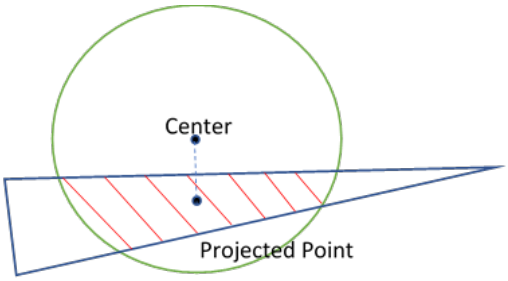
To know whether the triangle intersects with the sphere, we measure the distance between each vertex and the center of the sphere. If at least one distance is less than or equal to the radius of the sphere, then the triangle lies within the sphere as shown in Figure 35a.

Case 2 : A triangle whose vertices lie outside the sphere and the projection of the center of the sphere onto the plane containing the triangle lies within the triangle -

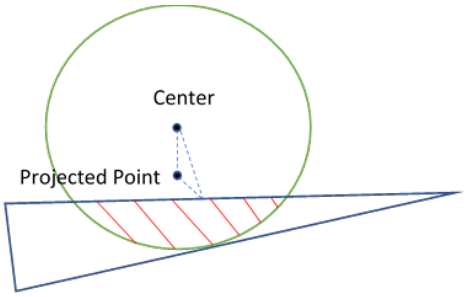
When the distance between the center and projected point is less than or equal to radius, then the triangle lies within the sphere as shown in Figure 35b.



(a) Case 1: Either of the three vertices are within the sphere.



(b) Case 2 : A triangle whose vertices lie outside the sphere and the projection of the center of the sphere onto the plane containing the triangle lies within triangle.



(c) Case 3 : A triangle whose vertices lie outside the sphere and the projection of the center of the sphere onto the plane containing the triangle lies outside triangle.

Figure 35: Three possible cases in which a triangle intersects the sphere.

Case 3 : A triangle whose vertices lie outside the sphere and the projection of the center of the sphere onto the plane containing the triangle lies outside the triangle -

The perpendicular distance[6] of center of the sphere to each of the three sides is calculated. If this perpendicular distance is less than or equal to the radius of the sphere, then the triangle lies within the sphere as shown in Figure 35c.

6.4 Backtracking Algorithm

Rather than performing ray tracing on every possible combination of reflecting triangles, it would be computationally efficient to remove the triangle combinations where reflection is impossible. This can be achieved through a backtracking algorithm [14].

[6]Using the Pythagoras theorem, the perpendicular distance of the center of the sphere from each of the sides can be calculated as the square root of the sum of square of projected distance and perpendicular distance of projected point to the side.

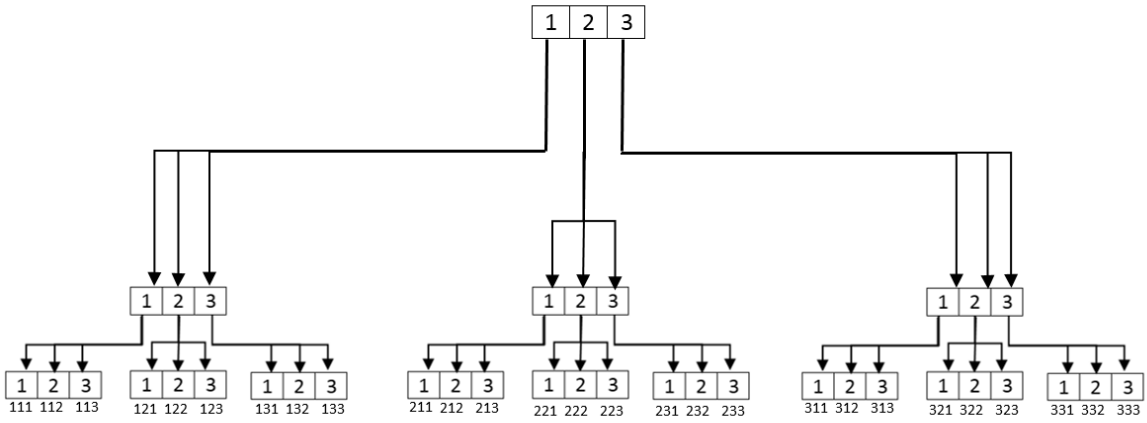


Figure 36: A recursive tree consists of three generations for a set 1,2,3. The last line shows all the combinations after traversing through every path in the tree.

Backtracking algorithm is used to find all the possible permutations by constructing a recursive tree. The set of all entities form the first generation of this tree, e.g., $\{1, 2, 3\}$ as shown in Figure 36. For every element in the first generation, the entire set becomes the second generation. This recursively takes place until the n-th generation of the tree. Every path that traverses from the first generation to the last generation gives one combination as shown at the bottom of Figure 36.

A reflection occurs if a ray is incident on the face of the triangle. For an n-th order reflection, the ray reflects n times from triangles (not necessarily distinct). All the triangle combinations are computed to find reflected rays. Tx is the first node of the tree and Rx is the last node of the tree. The entire set of triangles extracted from the CAD file is the second generation of the tree. Further, the child nodes of each element in the second generation are again represented by the complete set of triangles. This process of generating child nodes carries on n number of times that is equal to the order of reflection. In the last generation, all the nodes end at the Rx. An illustrative example to explain this backtracing method is described below: For every element in the first generation, the entire set becomes the second generation. This recursively takes place until the n-th generation of the tree. Every path that traverses from the first generation to the last generation gives one combination as shown at the bottom of Figure 36.

A reflection occurs if a ray is incident on the face of the triangle. For an n-th order reflection, the ray reflects n times from triangles (not necessarily distinct). All the triangle combinations are computed to find reflected rays. Tx is the first node of the tree and Rx is the last node of the tree. The entire set of triangles extracted from the CAD file is the second generation of the tree. Further, the child nodes of each element

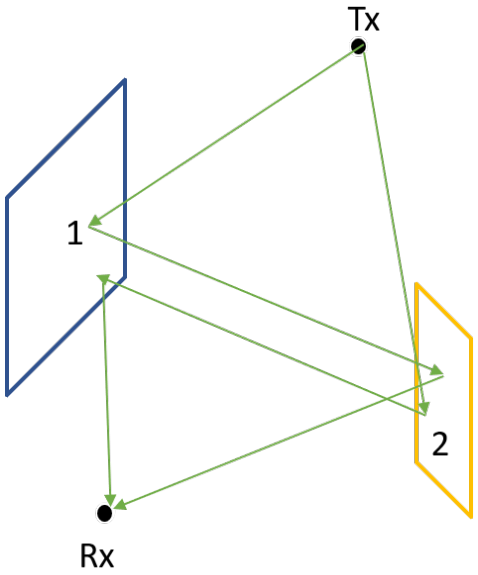


Figure 37: A scenario with 2 planes and a pair of Tx and Rx.

in the second generation are again represented by the complete set of triangles. This process of generating child nodes carries on n number of times that is equal to the order of reflection. In the last generation, all the nodes end at the Rx.

An illustrative example to explain this backtracking algorithm is described below:

1. Consider Tx as the first node or parent node of the tree as seen in Figure 37.
2. Consider Rx as the last node of the tree as seen in Figure 37.
3. The first generation of child nodes is the complete set of planes (CADOutput). The set of child nodes at each node of the first generation is the complete set of planes as seen in Figure 38.
4. A link does not exist between the identical nodes present in the first generation and the second generation, because a ray cannot consecutively reflect from the same plane. Such cases are eliminated as shown in Figure .37.
5. The final output is the set of all the combinations of the planes where reflection is possible.

Complexity: the worst-case time complexity is $\mathcal{O}(n^m)$ where n denotes the total number of planes and m represents the highest order of reflection.

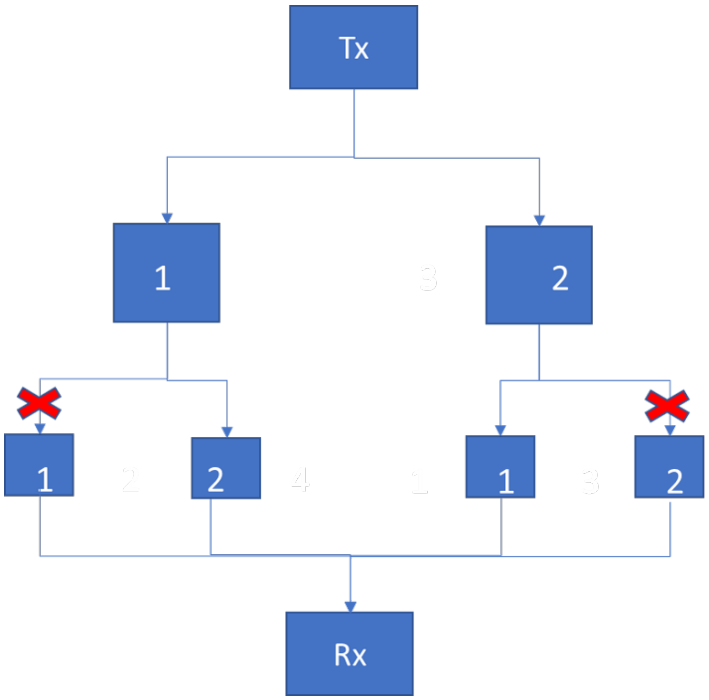


Figure 38: Schematic diagram of a Tree describing the backtracking algorithm.

6.4.1 Code Structure

Backtracking algorithm is implemented using `Treetraversal.m` and `VerifyTreeTraversalPath.m` functions. `CADOutput`, i.e., CAD information, generated from the previous step is given as input to the function `Treetraversal.m`. A loop is run to iterate through all `CADOutput` rows. as shown in flow chart in Figure 39. This means that we are iterating through all the triangles present in the CAD file. At every iteration, the possibility of ray is probed. If a ray does not exist, then we move onto the next iteration. If a ray can exist, then the variables `ArrayOfPlanes, ArrayOfPoints` and `ArrayOfMaterials` are populated. The variable `ArrayOfPlanes` contains the plane equations of the triangles, `ArrayOfPoints` contains the vertices of the triangles, and `ArrayOfMaterials` contains the material properties of the triangle. Once all the variables are populated, we recursively perform the same task for N times, where N denotes the order of reflection.

The row format of variables `ArrayOfPlanes`, `ArrayOfPoints` and `ArrayOfMaterials` are shown in the Figures 40, 41 and 42, respectively. $\{Tr1, Tr2, \cdots, TrN\}$ represent a set of triangle combinations, where N is the reflection order.

The parameters $\mathtt{x}_i$, $\mathtt{y}_i$, $\mathtt{z}_i$ are x, y, z coordinates of the i-th vertex of the triangle. From the vertices of the triangle, a plane equation is constructed and a plane equation is given as $ax + by + cz + d = 0$ where a, b, c are x, y, z coefficients, respectively, and d is a constant. `MaterialId` variable in `ArrayOfMaterials` helps in identifying the

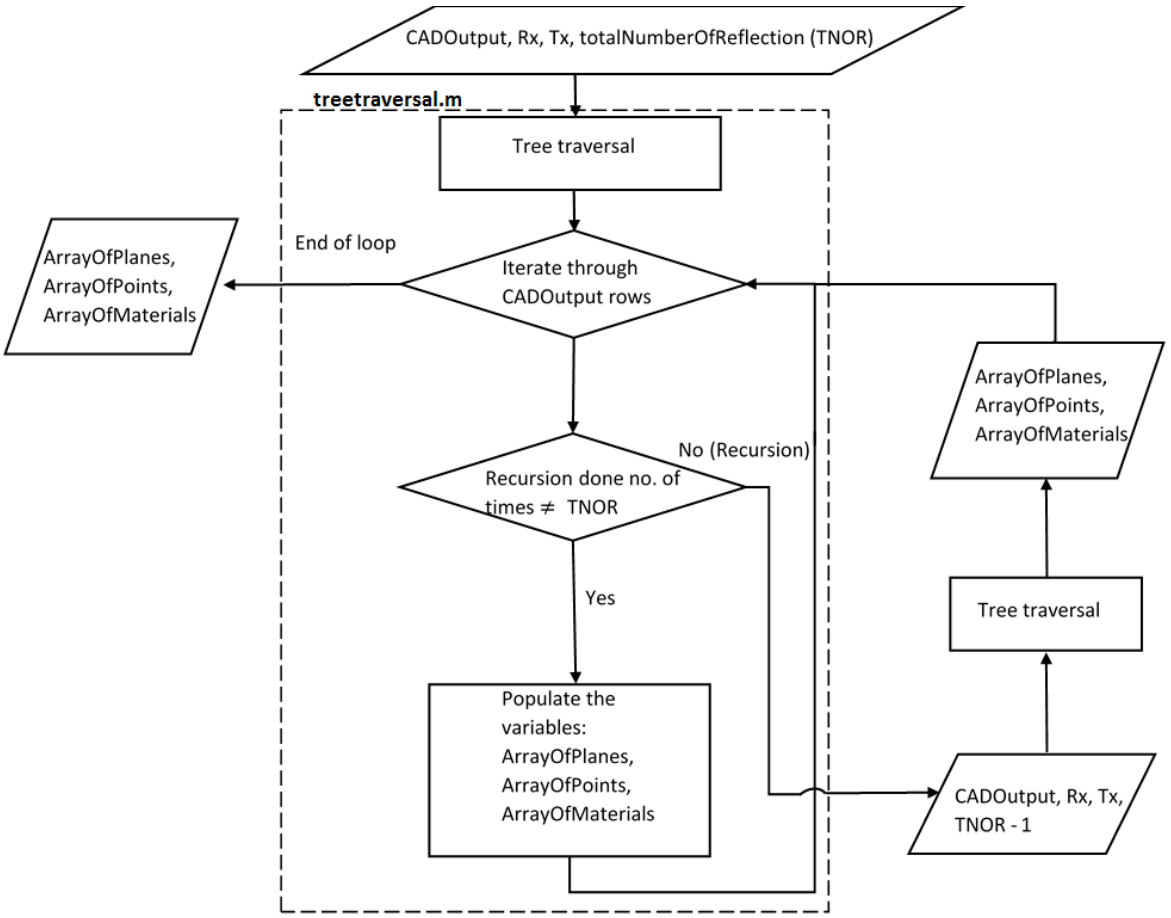


Figure 39: Flow chart of backtracking algorithm.

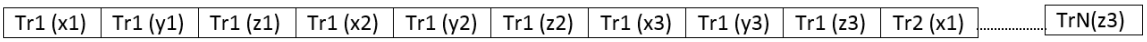


Figure 40: A row of the `ArrayOfPoints` variable.

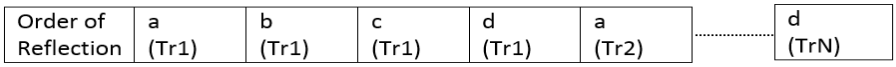


Figure 41: A row of the `ArrayOfPlanes` variable.

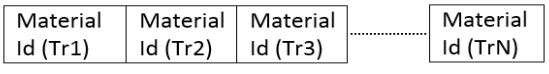


Figure 42: A row of the `ArrayOfMaterials` variable.

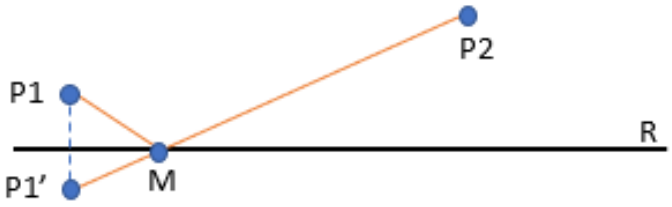


Figure 43: Reflected path between points $P1$, $P2$ and reflecting plane R.

material associated with the triangle.

6.5 Method of Images

The method of images is based on geometrical optics. According to the principles of geometrical optics, in free space a light ray travels in a straight line which has minimum path-length between two points. This phenomenon is known as *Fermat's principle* [15]. In case of reflection, the path-length of the ray should be minimum between two points while the ray is incident on a plane surface. This problem is known as *Heron's Problem* [16], which can be solved by taking the image $P1'$ of a point $P1$ on to the plane surface R and connecting the other point $P2$ to this image via a line as shown in Figure 43. The intersection point M of this line with the plane is again connected to the point $P1$.

Given a set of planes on which reflections are generated, obtained thanks to the backtracking algorithm, the reflection points can be computed using the method of images.

An example is presented below to explain the method of images to compute the second order reflection as shown in Figure 44.

1. Import Tx and Rx locations along with a set of planes as input parameters.
2. Compute image of Tx in Plane 1 which is the first recursion.
3. Compute the image of Tx image in plane 2, which is the second recursion.
4. Next, the vector joining the image of Tx image with Rx is calculated where the length of this vector is the path-length.
5. The intersection of this vector with plane 2 is the point of incidence on plane 2 for the reflected ray. By connecting the intersection with Rx, the DoA vector is obtained as shown in Figure 44.
6. A vector is formed by connecting the intersection of the ray with plane 2 and image of Tx. The intersection of this vector with plane 1 is subsequently computed.

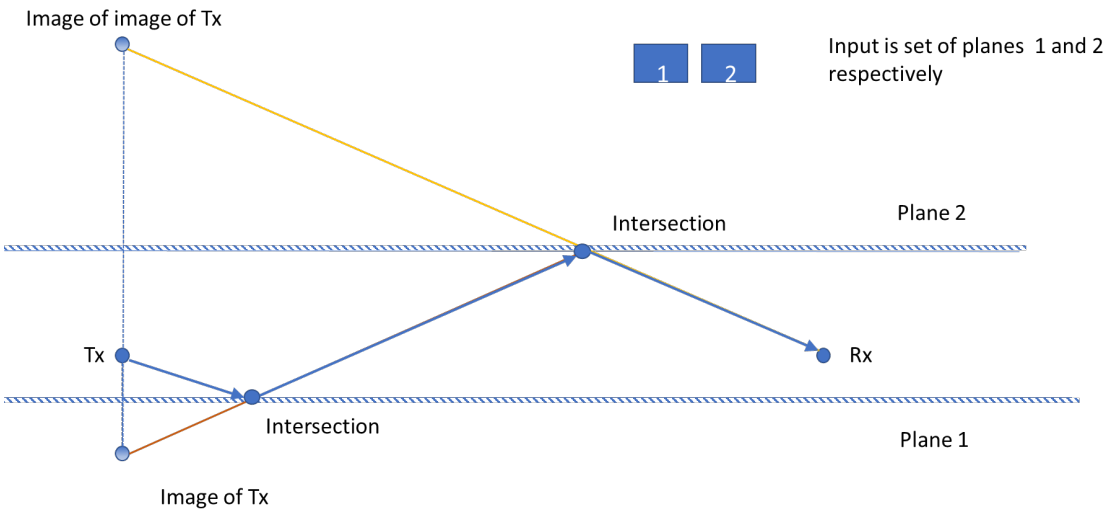


Figure 44: Method of images for a scenario with an input of two planes.

7. The intersection on plane 1 is connected to the intersection on plane 2, also forms a vector. This vector is a part of the reflected ray.

8. The Tx is connected to the intersection on plane 1. This gives the DoD vector.

As described in the backtracking algorithm, $\mathcal{O}(n^m)$ is the total time complexity to traverse through all the combinations of planes. In addition, there are total $(m+1)$ vectors generated through the method of images and each vector is verified for its intersection with any of the planes. This entire process takes $\mathcal{O}(n^{m+1})$ steps to complete. Therefore, the total complexity is $\mathcal{O}(n^{2m+1})$, which can be obtained by multiplying the complexities of both algorithms.

6.5.1 Code Structure

The method of images algorithm is implemented using the functions: `multipath.m`, `singlepathgenerator.m` and `LOSOutputgenerator.m`. In addition to these functions, `verifyPath.m` function is also used to verify the presence of the ray.

Note that `multipath.m` generates all the possible rays for given locations of PAAs at the transmitter and receiver (`Tx, Rx` in Figure 45) and order of reflection. It iterates through the output of tree traversal part of the code, i.e., `ArrayOfPlanes` and `ArrayOfPoints`. A single row of `ArrayOfPlanes` and `ArrayOfPoints` is passed to `singleMultipathGenerator.m` to generate a ray. Further, the Q-D model is applied to the ray generated through `singleMultipathGenerator.m`.

The method of images is implemented in `singleMultipathGenerator.m` which takes input from `multipath.m`. Next step is to extract the plane equations and images of `Tx` along these extracted planes. As shown in Figure 46, `singleMultipathGenerator.m` is called recursively for N times, where N is the order of reflection. The

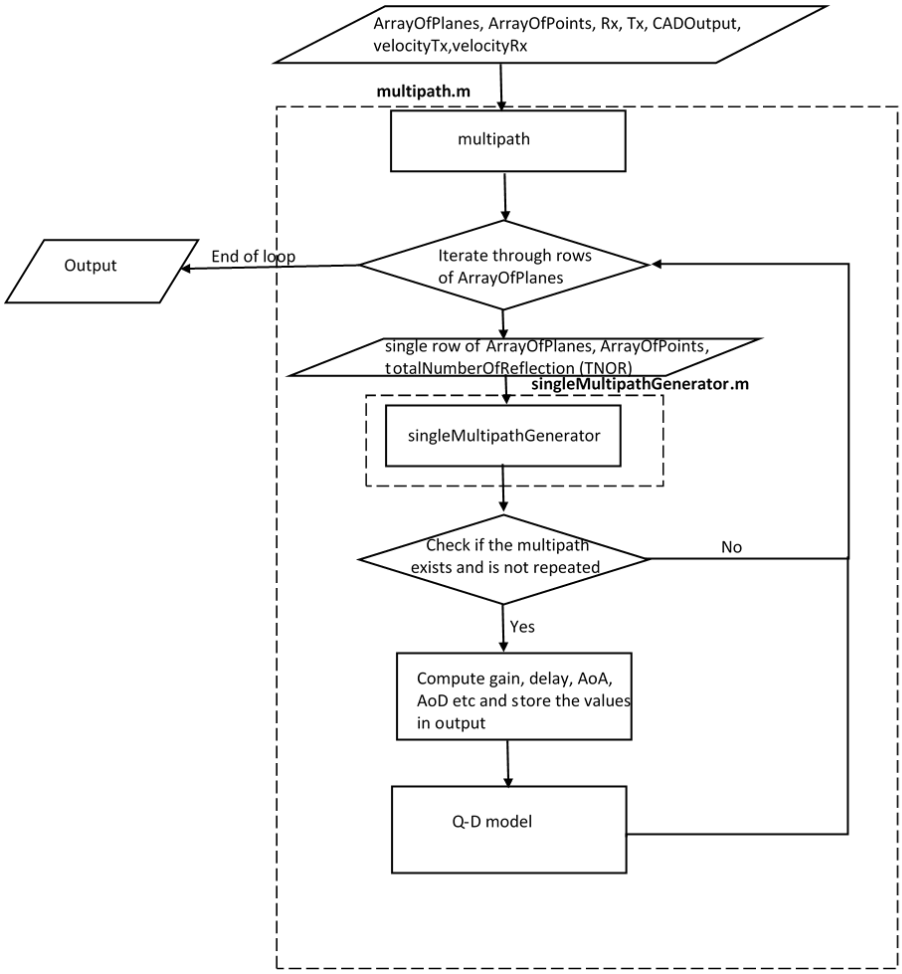


Figure 45: Flow chart of `multipath.m`.

final image of `Tx` is then connected to the `rx`. This forms the DoA vector. We also get path-length in this step.

In the next step, we compute the point where this DoA intersects with the plane where this reflection has happened. In the last recursion, DoD is assigned the value of DoA and when we exit from subsequent recursions, DoD is dynamically updated with the newly computed ray vectors which are a part of the reflected ray. We also check if DoD intersects with any other triangles in the CAD file using `verifyPath.m`. Note that the path between Tx and Rx exists when there is no intersection with any of the triangles. If the path exists, then we exit from this recursion and we enter into the function which recursively called the previous function. DoD is again calculated by joining the intersection point which was calculated in the previous call with the reflected image obtained in the present function. We again check if this DoD intersects with any of the other triangles or not. This process continues until there is an intersection. In the end, if there is no intersection with any of the triangles, the complete ray between

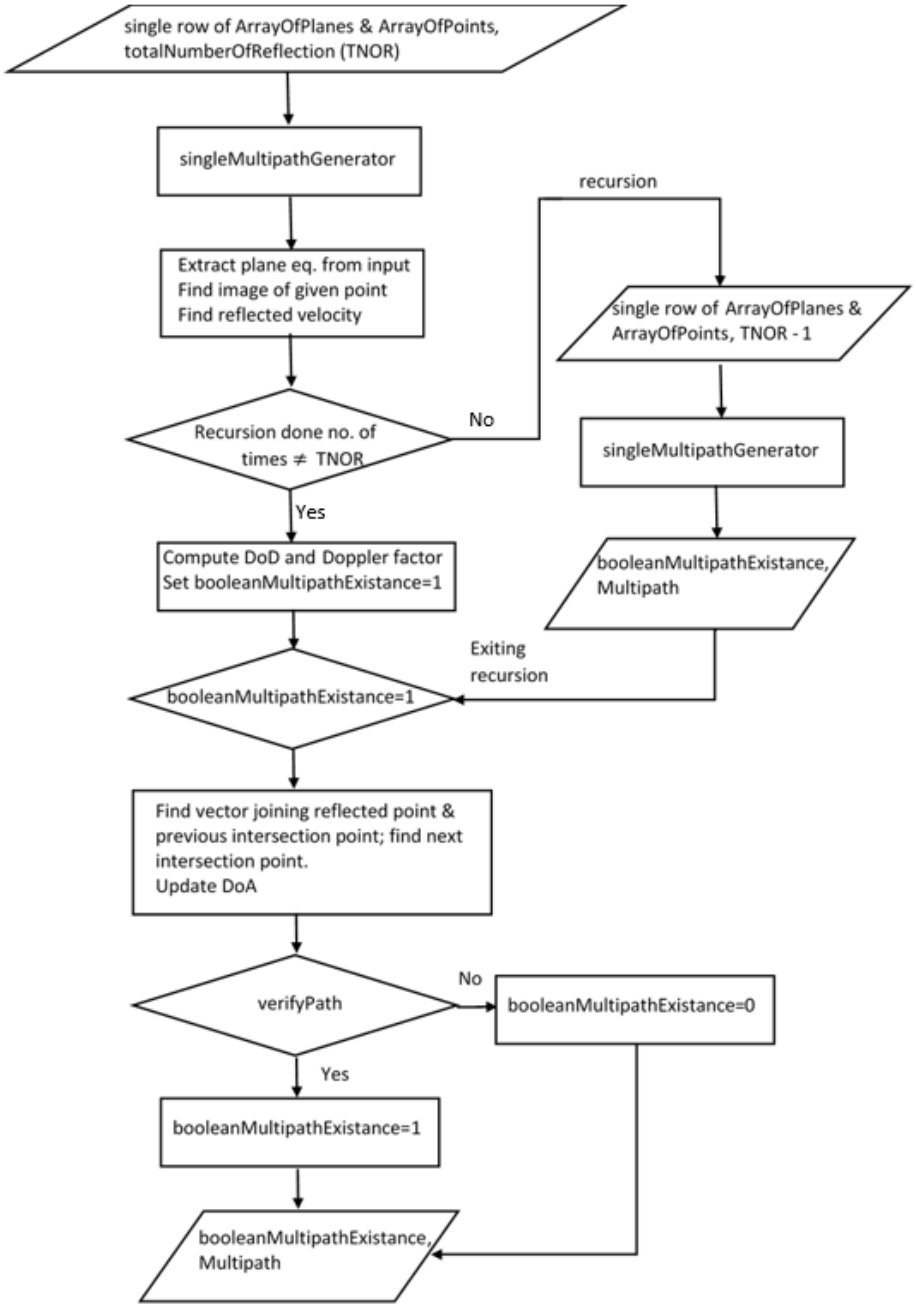


Figure 46: Flow chart of `singleMultipathGenerator.m`.

Tx and Rx is finally generated.

Based on the ray generated above, we compute gain, delay, AoA, AoD and phase in the global coordinates. The delay is obtained by dividing the path-length by the

Rx x	Rx y	Rx z	PoI 1 x	PoI 1 y	PoI 1 z	Tx x	Tx y	Tx z

Figure 47: `Multipath` variable for first order reflection, where POI denotes point of incident.

Rx x	Rx y	Rx z	PoI 2 x	PoI 2 y	PoI 2 z	PoI 1 x	PoI 1 y	PoI 1 z	Tx x	Tx y	Tx z

Figure 48: `Multipath` variable for second order reflection.

speed of light. The gain is computed by using Friis transmission formula (c.f. (1)). For every reflection, a phase shift of 180 degrees is added. Note that the phase of the ray changes by 180 degree when the ray reflects from the surface of a medium with higher refractive index than that of the medium in which it's traveling. The values computed are relative to a single transmitter-receiver PAA centroid and they are stored in the `Output` variable. `Output` is mapped into `OutputPAA`, that is a structure that stores the information of all the PAA centroid combinations.

While each PAA in the centroid experiences the same gain and delay, AoAs and AoD of each PAA might differ if they have a different orientation. For this reason the AoAs and AoD of each PAA are computed at the end of the ray tracing. The structure `frmRotMpInfo` is used to store AoA and AoD vectors as well as the transmitter and receiver velocity vectors in the global frame.

The Rx, points of intersection and Tx of a ray are stored in a variable named `multipath`.Multipath variable is a 2D array. Each row represents a set of points of a given multipath component. By joining these points, we can reconstruct the MPC. The points are arranged from Rx, last point of incidence, second last point of incidence to first point of incidence and then Tx. For first order reflection, this would be Rx, first point of incidence (PoI 1) and then Tx as shown in the Figure 47. For second order reflection, this would be Rx, second point of incidence (PoI 2), first point of incidence (PoI 1) and then Tx as shown in the Figure 48. Note that `LOSOutputgenerator.m` is a simple function which computes the LOS path for a given Tx and Rx locations. This is done by connecting `Tx` and `Rx`, and checking if it does not pass through any of the obstacles using `verifyPath.m`. It is similar to `singleMultipathGenerator.m` and is implemented in `Raytracer.m` function.

References

[1] C. Lai, R. Sun, C. Gentile, P. B. Papazian, J. Wang, and J. Senic, "Methodology for multipath-component tracking in millimeter-wave channel modeling," *IEEE Transactions on Antennas and Propagation*, vol. 67, no. 3, pp. 1826–1836, March 2019.

[2] A. Maltsev, A. Pudeyev, A. Lomayev, and I. Bolotin, "Channel models for ieee 802.11 ay," *IEEE doc*, pp. 802–11, 2017.

[3] M. Lecci, M. Polese, C. Lai, J. Wang, C. Gentile, N. Golmie, and M. Zorzi, "Quasi-deterministic channel model for mmwaves: Mathematical formalization and validation," *arXiv preprint arXiv:2006.01235*, 2020.

[4] A. Maltsev, A. Pudeyev, I. Karls, I. Bolotin, G. Morozov, R. Weiler, M. Peter, and W. Keusgen, "Quasi-deterministic approach to mmwave channel modeling in a non-stationary environment," in *2014 IEEE Globecom Workshops (GC Wkshps)*, Dec 2014, pp. 966–971.

[5] M. Born and E. Wolf, *Principles of optics*. Cambridge University, 2009.

[6] S. Blandino, G. Mangraviti, C. Desset, A. Bourdoux, P. Wambacq, and S. Pollin, "Multi-user hybrid mimo at 60 ghz using 16-antenna transmitters," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 2, pp. 848–858, 2019.

[7] A. Hughes, C. G. S. Yun Jun, D. Caudill, J. Chuang, and J. Senic, "Impact of beamwidth on the coherence distance of indoor and outdoor 60GHz measured channels," *NIST internal report*, 2020.

[8] J. B. Kuipers, *Quaternions and rotation sequences: a primer with applications to orbits, aerospace, and virtual reality*. Princeton university press, 1999.

[9] G. Durgin, N. Patwari, and T. S. Rappaport, "Improved 3d ray launching method for wireless propagation prediction," *Electronics Letters*, vol. 33, no. 16, pp. 1412–1413, July 1997.

[10] H. Lipson, "Amf tutorial: The basics (part 1)," *3D Printing and Additive Manufacturing*, vol. 1, pp. 85–87, 06 2014.

[11] H. Assasa, J. Widmer, T. Ropitault, A. Bodi, and N. Golmie, "High Fidelity Simulation of IEEE 802.11ad in ns-3 Using a Quasi-deterministic Channel Model," in *Workshop on Next-Generation Wireless with ns-3*, ser. WNGW '19, Florence, Italy, 2019.

[12] C. Gentile, P. B. Papazian, R. Sun, J. Senic, and J. Wang, "Quasi-deterministic channel model parameters for a data center at 60 ghz," *IEEE Antennas and Wireless Propagation Letters*, vol. 17, no. 5, pp. 808–812, 2018.

[13] R. Charbonnier, C. Lai, T. Tenoux, D. Caudill, G. Gougeon, J. Senic, C. Gentile, Y. Corre, J. Chuang, and N. Golmie, "Calibration of ray-tracing with diffuse scattering against 28-ghz directional urban channel measurements," *IEEE Transactions on Vehicular Technology*, 2020.

[14] S. Dasgupta, C. H. Papadimitriou, and U. V. Vazirani, *Algorithms*. McGraw-Hill Higher Education, 2008.

[15] M. Born and E. Wolf, *Principles of optics*. Cambridge University, 2009.

[16] L. Figueiras and J. Deulofeu, "Visualising and conjecturing solutions for heron's problem," *Proceedings of the CERME 4 international conference*, p. 420–427, 2005.