5G MIMO Data for Machine Learning: Application to Beam-Selection using Deep Learning

Aldebaro Klautau, Pedro Batista,
Dep. of Comp. and Telecomm. Eng.
Federal University of Pará
Belem, PA, 66075-110, Brazil
Emails: {aldebaro,pedro}@ufpa.br

Nuria González-Prelcic,
Dep. of Signal Theory and Comm.
University of Vigo
Vigo, 36310, Spain
Email: nuria@gts.uvigo.es

Yuyang Wang and Robert W. Heath Jr.
Dep. of Elec. and Computer Eng.
The University of Texas, Austin
Austin, TX, 78712-1084, USA
Emails: {yuywang,rheath}@utexas.edu

***Abstract*— The increasing complexity of configuring cellular networks suggests that machine learning (ML) can effectively improve 5G technologies. Deep learning has proven successful in ML tasks such as speech processing and computational vision, with a performance that scales with the amount of available data. The lack of large datasets inhibits the flourish of deep learning applications in wireless communications. This paper presents a methodology that combines a vehicle traffic simulator with a ray-tracing simulator, to generate channel realizations representing 5G scenarios with mobility of both transceivers and objects. The paper then describes a specific dataset for investigating beam-selection techniques on vehicle-to-infrastructure using millimeter waves. Experiments using deep learning in classification, regression and reinforcement learning problems illustrate the use of datasets generated with the proposed methodology.**

I. Introduction

Machine learning (ML) has been applied to a large variety of problems in telecommunications, which include network management, self-organization, self-healing and physical layer (PHY) optimizations [1], [2]. Deep learning (DL), a special category of ML, has been responsible for several recent performance breakthroughs in areas such as speech processing and computational vision [3]. The success in other domains motivates the application of DL to communication problems [4]–[12]. While DL can be applied to any ML problem, its niche has been applications with large amount of data. The reason is that DL scales well with the amount of data and model complexity [3].

In many DL application domains, the data is abundant or has a relative low cost. For example, the DL-based text-to-speech system presented in [13], which represents the state-of-the-art, achieves quality close to natural human speech after being trained with a reasonable amount of digitized speech. In contrast, the research and development of 5G lower layers has to deal with a relatively limited amount of data. For example, mmWave measurements for 5G MIMO research demands very expensive equipment and, eventually, elaborate outdoor measurement campaigns [14]. The lack of freely available data impairs the data-driven lines of investigation.

This paper presents a methodology for channel data generation in 5G millimeter wave (mmWave) multiple-input multiple-output (MIMO) scenarios [15]. The goal is to facilitate the investigation of ML-based problems related to the PHY of mmWave MIMO in 5G. The presented methodology simplifies creating data in complicated (and potentially realistic) mobility scenarios through repeatedly invoking a traffic simulator and a ray-tracing simulator. In the current context, generating propagation channel data is a reasonable way to alleviate the data scarcity while benefiting from the accuracy associated to ray-tracing (RT) [16], [17]. For instance, RT can cope with 5G requirements such as *spatial consistency*, which has been a challenge to traditional stochastic modeling [18], [19]. The simulated datasets do not substitute but complement data from measurements, which can improve and validate simulated data and statistical channel models, as they become available. The paper also presents concrete examples of usage for the generated datasets via experiments with DL for beam-selection in vehicle-to-infrastructure (V2I) mmWave communications. Given the current amount of data is limited, it is out of the scope of this paper to investigate performance of specific DL architectures. The goal is to illustrate instead the flexibility provided by the data generation methodology. This methodology can be used in applications other than V2I, as well as to create datasets for ML problems concerning classification, regression, clustering and time-based *sequence* recognition [20].

Several ML techniques have been applied to the PHY processing (see, e. g., [21] and references therein). The large majority of previous work rely on simulations. For example, simulations are used in [22], which present unsupervised DL architectures based on autoencoders for MIMO schemes. In [23], a classifier based on boosted trees was applied to the optimization of handovers between sub-6 GHz and mmWave radios using simulated channels. In such cases, in which measurements are not available, our methodology can provide reasonably accurate channel data and advanced tools for modeling mobility. The generated datasets are especially useful when spatial consistency and time evolution are important to assess the ML technique.

The rest of the paper is organized as follows. The methodology for data generation is presented in Section II. Beam-selection is the topic of Section III, in which a brief literature review is included. We discuss experiments illustrating deep learning using the V2I dataset in Section IV, which is followed by the conclusions in Section V.

II. Methodology for Data Generation

In this section we describe the proposed methodology for data generation, which has to take in account the challenges of mmWave channel modeling such as the large signal bandwidths and prominent impact of scattering [16]. Other issues raise due to the intended application in 5G MIMO. In scenarios of high mobility such as V2I, the channel evolution over time is important to assess, for example, beam tracking techniques [24]. For instance, mmWave communication when the vehicle speed is 35 m/s may have to cope with a signal fading rate of 44 dB/s [14]. In fact, difficult channel modeling in complex scenarios is the first issue highlighted in [21] when discussing challenges related to DL in wireless communications. The following paragraphs discuss how RT and traffic simulators are used in our methodology for circumventing issues when generating ML datasets that depend on mmWave MIMO modeling.

A. Ray-tracing simulation for mmWave MIMO channels

RT is considered a promising simulation strategy for 5G (see, e. g. [17]). RT can provide very accurate results [16], [25] but its computational cost increases exponentially with the maximum allowed number of reflections and diffractions [26]. Another issue of RT is that the generated channels are *site specific*, depending on the specific propagation environment. Besides, for improved RT accuracy, the scenario should be reasonably detailed. For example, outdoor scenarios require the detailed specification (including size, geometry, material, etc.) of buildings, vehicles, people and objects of interest such as a roadside unit (RSU) for V2I, as illustrated in Fig. 1. The geometrical aspects of the environment must be informed together with the corresponding electromagnetic parameters such as the *scattering coefficient* (S) for each material [17]. Given the simulation scenario, a RT simulator projects rays in the three-dimensional angular space with a predefined spacing. Then, the paths are ranked according to the received power of each ray. A detailed enough scenario description is the first challenge for RT usage. Another one is an appropriate modeling of the propagation channel, which has to take in account, for example, the scattering of mmWave signals.

Diffuse scattering (DS) is an important feature of mmWave channel simulators [17], [27], [28]. This feature can enrich the channel realizations and minimize the chances of bias due to a limited number of specular rays, as found when materials are smooth. The computational cost increases though with parameters such as the maximum allowed number of DS reflections ($N_{\text{max}}^{\text{DS}}$). To illustrate DS, Fig. 2 shows an example[1] of rays obtained in a simulation with a traffic jam (vehicles with receivers are marked in red) using 60 GHz. In this case, the simulation time increased by a factor of three when enabling DS with $N_{\text{max}}^{\text{DS}} = 2$.

A RT simulator may support mobility, for example, allowing the receiver to follow a trajectory with a given speed. Supporting the changing position of scatterers and blockers, though, complicates the required RT optimizations. Therefore, it is more common for a RT simulator to allow receivers to move, but not objects that can influence the rays. Simulating mobility then requires repeatedly invoking the simulator with the specification of a *scene*. This is the case of Remcom's Wireless InSite [28], which is the RT simulator used in this paper.

[1]For visualization, the camera in Fig. 2 is rotated with respect to Fig. 1.

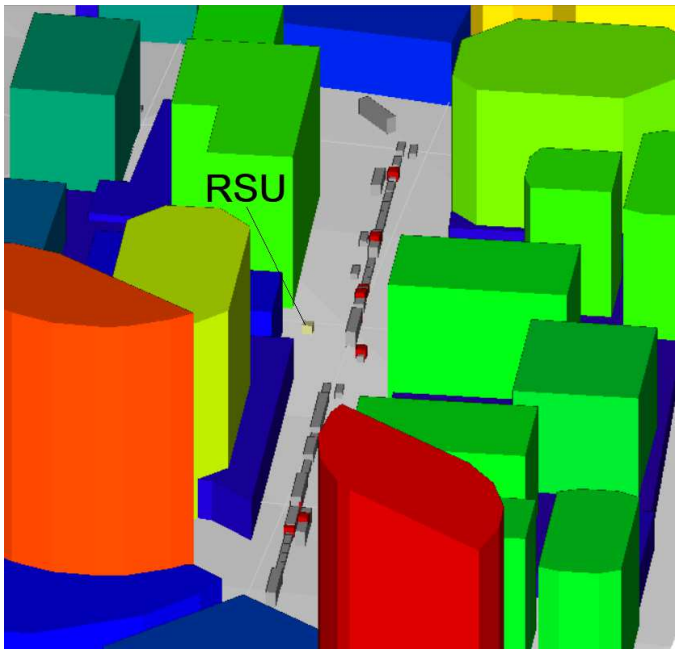

Fig. 1. Urban canyon scenario in a 3-d ray-tracing simulator with vehicles of distinct sizes randomly positioned. The building color indicates height and corresponds to a range from 0 (blue) to 101 meters (red).

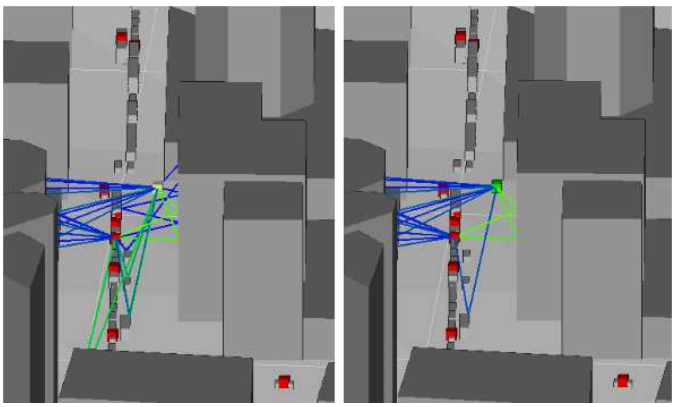

Fig. 2. Rays obtained in a traffic jam situation. The left figure shows all 25 most significant rays reaching a vehicle while the right one shows the subset (8 rays) corresponding to the "diffuse scattered" rays. The zoomed figure indicates there are three clusters of diffused rays reaching this receiver.

B. Spatial consistency and time evolution requirements

While traditional *drop-based* stochastic models have been extremely useful in the design of communication systems, their application in the context of 5G has been criticized with respect to spatial consistency [18], [29]. For example,

the results presented in [19] indicate that the 3GPP three-dimensional (3-d) geometry-based stochastic channel model underestimates the performance of massive MIMO systems in line-of-sight (LOS) scenarios, while overestimating the performance of multi-user MIMO in specific ultra-dense scenarios with non-LOS (NLOS).

State-of-the-art stochastic and hybrid models for 5G have been incorporating features that aim at improved spatial consistency [30], [31]. Examples are the modeling techniques in NYUSIM [32] and QuaDRiga [33], as well the three techniques detailed in [34]. A classical alternative to stochastic models is RT [17], [35], which is the simulation technique adopted in this paper. RT is able to generate data with two key requirements for ML datasets with simulated channel realizations: spatial consistency and history of time evolution.

In our methodology, the outputs of the simulators are periodically stored as "snapshots" (or *scenes*) over time $t = nT_{\text{sam}}$, where T_{sam} is the sampling period and $n \in \mathbb{Z}$. A *scene* $\mathcal{S}(t)$ can contain multiple transmitters and receivers, which facilitates using the datasets to investigate multiuser and other MIMO problems. In post-processing stages (an example is provided in Section III-D), the information in $\mathcal{S}(t)$ is used to model, for example, MIMO channels $\mathbf{H}(t)$. A scene $\mathcal{S}(t)$ can potentially contain all *out-of-band* information that the user gathered, including the ones provided by the RT and traffic simulators, such as position, vehicle dimension, angles of arrival, gains, etc. Section IV will discuss concrete examples on how the information in $\mathcal{S}(t)$ can be used in beamforming applications.

For improved scene diversity and given the relatively high computational cost of a RT simulation, we extract observation windows (or *episodes*) at distinct instants. Specifically, instead of always consecutively extracting a scene along the whole simulation, episodes of duration T_{epi} are obtained, each with $N_{\text{sce}} = \lfloor T_{\text{epi}}/T_{\text{sam}} \rfloor$ scenes. For example, an episode starting at time t_0 will be composed by a sequence of scenes $\{\mathcal{S}(t), t = t_0, t_0 + T_{\text{sam}}, \ldots, t_0 + (N_{\text{sce}} - 1)T_{\text{sam}}\}$. For facilitating parallel processing, a dataset with N_{epi} episodes can be organized as a TensorFlow TFRecord [20].

Keeping the channel variation over time enables investigating algorithms that take in account the channel dynamics. For more realistic simulations, the mobility can be controlled by a specialized software as described in the next subsection.

C. Integration of traffic and ray-tracing simulators

Vehicle and pedestrian *traffic simulators* provide flexibility to investigate the impact of mobility in V2I and related applications. We describe the integration between the open source *Simulation of Urban MObility* (SUMO) traffic simulator [36] and Wireless InSite. There is extensive support to the use of SUMO with network simulators such as OMNeT++, but the novel integration with RT facilitates studies targeting the mmWave PHY.

The main role of the traffic simulator is to facilitate modeling mobility, especially the motions of both transceivers and potential scatters in the environment. It is possible to directly get the necessary data only with the RT simulator but this may require considerable effort if the scenario is complicated. Traffic simulators are specialized tools with plenty of features to describe vehicles with distinct characteristics, interaction with pedestrians, etc. Adopting the right tool to model mobility enables the user to depart from simplistic scenarios, such as those in which all vehicles have constant speed. Using a specialized traffic simulation tool to decouple the mobility specification from the RT simulation, simplifies the experiment configuration and grants to the user flexibility, for instance, to impose trajectories to any object or person, use distinct speeds, etc. Also, the orientation of objects such as antennas can be automatically adjusted.

To support our methodology, we wrote a Python *orchestrator* code to repeatedly invoke the traffic simulator, convert the vehicles position to a format that can be interpreted by the RT simulator, invoke the latter and post-processing the RT results to create episodes, as depicted in Fig. 3.

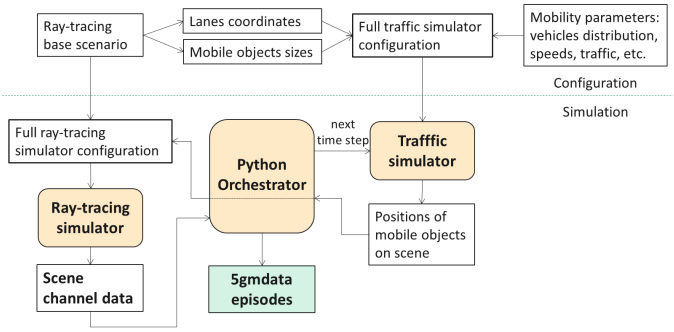

Fig. 3. Methodology that integrates ray-tracing and traffic simulators.

The main steps of the proposed methodology can be organized into *configuration* and *simulation* stages. In the configuration stage (upper blocks in Fig. 3), the user provides, e. g., information to enable conversion of coordinates between the two main softwares. Fig. 4 illustrates how the streets of interest in Fig. 1 are represented in the traffic simulator after having the coordinates properly converted. To facilitate the interaction with the traffic simulator, the orchestrator associates each mobile transmitter or receiver to a mobile object (MOBJ). A MOBJ can also simply play the role of a blocker or scatterer, with no associated transceiver. In the configuration stage, for each episode, the user specifies the *base scenario* files. The base scenario files, together with positions for all MOBJs, specified by the traffic simulator, compose all information required for a complete RT simulation. For simplicity, it is assumed in this paper that all episodes are generated with the same base scenario.

In the simulation stage, the orchestrator invokes the traffic simulator and then positions the MOBJs to compose a scene. Based on the output of the traffic simulator, some files of the base scenario are modified and stored in a unique folder. For each scene, this folder path is stored in order to allow reproducing the RT simulation of that scene. This enables the user to later extract additional information through customized software routines, as well as visualize results for a

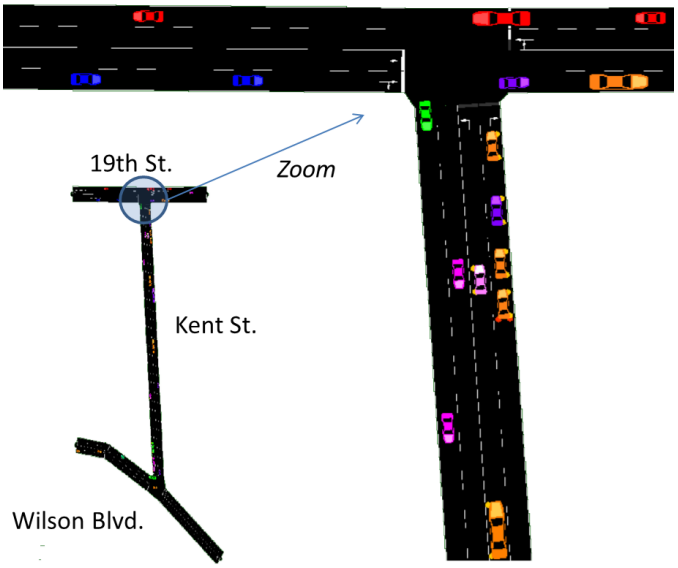

Fig. 4. Streets of interest for RT in Fig. 1 as represented in the traffic simulator. The junction at the right corresponds to a zoom of the intersection between Kent and 19th streets.

scene using the RT and traffic simulator's GUIs. Similarly, the corresponding information about the traffic simulation is recorded. For instance, this allows to retrieve the positions (x, y, z) and dimensions (l, w, h) of all MOBJs in a given time instant.

The steps of our methodology are summarized as follows:

Configuration:

- *Ray-tracing simulator*
  - Define base scenario "city", "terrain", etc. importing from geographic information systems (GIS) or using computer-aided design (CAD) software
  - Specify coordinates for RT "study area" and "mobility lanes" for cars, pedestrians, etc.
  - Create 3-d MOBJs and specify their electromagnetic properties
- *Traffic simulator*
  - Import from ray-tracing configuration the lanes coordinates and sizes of MOBJs
  - Specify MOBJs distribution, routes, traffic statistics, maximum speeds, accelerations, etc.

Simulation:

- *Python orchestrator code*, repeatedly:
  - Randomly selects the start of an episode
  - With sampling period T_{sam}, for each scene:
    - Invoke the traffic simulator and get the positions of all MOBJs
    - Create a full configuration for the ray-tracing simulator and execute it
  - Retrieve the ray-tracing outputs and organize them as episodes.

Following the steps of our methodology lead to the creation of simulation data of 5G **mm**Wave **MIMO** systems involving **m**obility (or *5GMdata*) that can be used in different applications. In summary, the dataset stores for each episode: base scenario folder and traffic simulator configuration file paths, episode start time, sampling period T_{sam}, number of transmitters, receivers and MOBJs, dimensions of all MOBJs, mappings between transmitters / receivers and MOBJs, coordinates of the *RT study area* and number L of rays per transmitter / receiver pair. Besides, the episode contains information for all its scenes. For a given scene, the information collected from the outputs of the RT and traffic simulators for each transmitter / receiver pair (m, n) are: average time of arrival $\overline{\tau}_{mn}$ (the subscripts mn will be omitted hereafter), total transmitted $\hat{P}_{\text{tx}}$ and received $\hat{P}_{\text{rx}}$ powers, and for the ℓ-th ray, $\ell = 1, \ldots, L$, complex channel gain α_ℓ, time of arrival τ_ℓ, angles ϕ_ℓ^D, θ_ℓ^D, ϕ_ℓ^A, θ_ℓ^A, corresponding respectively to azimuth and elevation for departure and arrival. Besides, a string s_ℓ stores all ray "interactions" (reflection, diffraction, DS) and facilitates distinguishing LOS and NLOS situations.

After 5GMdata is obtained, additional post-processing stages can generate the required data for specific target applications. For concreteness, the next section discusses possible uses of 5GMdata in the V2I context.

III. Machine Learning for Beam-Selection in V2I

In this section, we illustrate the application of our proposed framework. Specifically, we generate data for the application of ML to predict the best beam pairs in the context of mmWave to cellular systems (the V2I setting).

A. Brief literature review of beam-selection

MmWave MIMO is a means to exchange sensor data in vehicular systems [37]. A main challenge is that mmWave, as initially envisioned for this application, requires the pointing of narrow beams at both the transmitter and receiver. Taking into account extra information such as out-of-band measurements and vehicles positions can reduce the time needed to find the best beam pair [37]–[39]. Beam training is part of standards such as IEEE 802.11ad and 5G, and has also been extensively studied in the context of wireless personal and local area networks (see, e. g. [40], [41] and references therein). Among the several problems related to beam training and tracking in distinct scenarios [39], this paper focuses on a subset collectively called *beam-selection* in V2I. The goal is to choose the best pair of beams for analog beamforming, with both transmitter and receiver having antenna arrays with only one radio frequency (RF) chain and fixed beam codebooks. The next subsection describes the information extracted for V2I ML experiments.

B. Dataset for machine learning in V2I

Using our methodology, 5GMdata is organized with the following characteristics. We generated all episodes with the base scenario depicted in Fig. 1, which corresponds to a 3-d model that is part of Wireless InSite's examples. The

scenario represents a region of Rosslyn,[2] Virginia, which was studied e. g. in [42]. The RT area of study is a rectangle of approximately 337×202 m^2. A transmitter is located at the RSU on Kent Street, as depicted in Fig. 1. We placed receivers on top of 10 vehicles (some identified in red in Fig. 1) and obtained 50 scenes per episode. The experiments reported in this paper concern 116 episodes. Wireless InSite's command-line *wibatch* is adopted for its support to the X3D model, which implements DS. Table I describes the most important simulation parameters.

TABLE I
SIMULATION PARAMETERS.

Ray-tracing parameters	
Carrier frequency	60 GHz
RSU transmitted power	0 dBm
RSU antenna height	5 m
Antenna (Tx and Rx)	Half-wave dipole
Propagation model	X3D
Terrain and city material	ITU concrete 60 GHz
Vehicle material	Metal
Ray spacing (degrees)	1
Num. L of strongest rays	25
Diffuse scattering model	Lambertian
DS max. reflections ($N_{\text{max}}^{\text{DS}}$)	2
DS coefficients (S)	0.4 (concrete), 0.2 (metal)
Traffic parameters	
Number of lanes	4
Vehicles	car, truck, bus
Lengths, respectively (m)	4.645, 12.5, 9.0
Heights, respectively (m)	1.59, 4.3, 3.2
Probabilities, respectively	0.7, 0.1, 0.2
Average speed (m/s)	8.2
Sampling period T_{sam} (s)	0.1

After the 5GMdata is obtained, the following post-processing is adopted.

C. Machine learning input features

The ML problems illustrated in this paper address beam-selection based solely on vehicles positions and sizes. It is assumed that the RSU receives through an error-free channel the position and a unique index of all vehicles for each scene. Based on the vehicle index, the RSU knows its dimension and may incorporate it as extra out-of-band information provided to the ML algorithm. The position and identity information is then represented as a matrix. Based on this input, a ML algorithm should estimate parameters of interest to beam-selection.

The *V2I study area*, with 23×250 m^2, is a subarea of the RT study area consisting basically of the street in which the RSU is located. A grid with resolution of 1×1 m^2 is adopted to represent the V2I study area, leading to a matrix $\mathbf{Q}_s$ of dimension 23×250 to represent each scene s. The RSU has a fixed position, which is then not explicitly specified in $\mathbf{Q}_s$. A negative element in $\mathbf{Q}_s$ indicates that the corresponding location is occupied (even partially) by a vehicle that is not a receiver or transmitter. The magnitude of this negative value indicates the vehicle's height. For example, -1 and -2 represent a car and a truck, respectively. A positive integer value r in $\mathbf{Q}_s$ represents the location of the r-th receiver, while 0 denotes the position is not occupied. Fig. 5 illustrates an example where the receiver is blue and the surrounding vehicles are yellow.

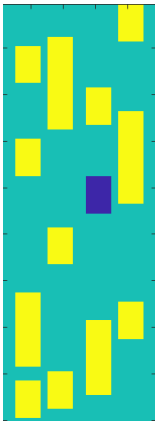

Fig. 5. Image corresponding to an example of an input features matrix $\mathbf{Q}_s$ representing 13 vehicles in four (vertical) lanes. The elements of $\mathbf{Q}_s$ corresponding to the pixels of the receiver (blue vehicle) are $+1$ while elements with value -1 (yellow) identify the positions of the other vehicles.

The next subsection describes post-processing schemes to extract information useful for generating the outputs of ML problems.

D. Post-processing ray-tracing outputs

The definition of desired beam-selection outcomes requires to model the composition of channels and beams based on the RT outputs. The estimated beams allow, for instance, the definition of *optimal* beam pair indices to be used as target outputs (or *labels*) in *supervised learning* [20]. The beams are assumed to have a beamwidth of β radians. We consider two different channel models in the next paragraphs, but others can be adopted such as *wideband* models (see, e. g., [15] and references therein). The first case is called *mmWave massive MIMO* model and considers each ray as a beam. The other case represents a more realistic situation that assumes fixed beam codebooks at transmitter and receiver, and uses basic signal processing techniques to obtain the received power for beam pairs.

1) MmWave Massive MIMO: In the massive MIMO case, the number N of antenna elements is large, and assuming $N \to \infty$ implies a small beamwidth $\beta \to 0$. Therefore, in this model, the departure $[\phi_{\ell_*}^D, \theta_{\ell_*}^D]$ and $[\phi_{\ell_*}^A, \theta_{\ell_*}^A]$ arrival directions of the *strongest* ray ℓ_* indicate the target *optimal* angles, which can be used in regression problems. In the case

[2]The lanes adopted in this paper follow the 3-d model geometry but do not actually exist.

of interest in classification problems, one can quantize the angles $[\phi^D_{\ell_*}, \theta^D_{\ell_*}, \phi^A_{\ell_*}, \theta^A_{\ell_*}]$ using vector or scalar quantization. If the latter is used, the angles can be quantized into four indices $[D_{\text{azi}}, D_{\text{ele}}, A_{\text{azi}}, A_{\text{ele}}]$ according to their dynamic ranges in the training set. These indices can be eventually converted to a single label for traditional classification. Typically, due to the scenario geometry, the number M of unique vectors that occur in the dataset is smaller than the total number of Cartesian products among $[D_{\text{azi}}, D_{\text{ele}}, A_{\text{azi}}, A_{\text{ele}}]$. It is therefore useful to pre-process the quantized values and map the vectors that actually appeared in the data into the range $\{1, 2, \cdots, M\}$, where M is the number of *class labels*. When training classifiers, one can then conveniently represent the labels with *one-hot* encoding to facilitate training neural networks, for example [20].

2) Codebook-based beams: In practical mmWave systems, N is finite and influences the beamwidth $\beta_{\text{N}} > 0$ for the projected beam given the antenna arrays. To take this in account, one can estimate the MIMO channel by combining the RT output with the mmWave *geometric channel model* as follows (see, e. g., [39] and references therein):

$$\mathbf{H}_{mn} = \sqrt{N_t N_r} \sum_{\ell=1}^{L} \alpha_\ell \mathbf{a}_r(\phi^A_\ell, \theta^A_\ell) \mathbf{a}^*_t(\phi^D_\ell, \theta^D_\ell), \qquad (1)$$

where N_t and N_r are the numbers of antennas at the n-th transmitter and m-th receiver, α_ℓ is the complex channel gain, $\mathbf{a}_r(\phi^A_\ell, \theta^A_\ell)$ and $\mathbf{a}^*_t(\phi^D_\ell, \theta^D_\ell)$ are the steering vectors at the receiver and transmitter for the ℓ-th path, respectively. We also assume DFT codebooks $\mathcal{C}_t = \{\bar{\mathbf{w}}_1, \cdots, \bar{\mathbf{w}}_{|\mathcal{C}_t|}\}$ and $\mathcal{C}_r = \{\bar{\mathbf{f}}_1, \cdots, \bar{\mathbf{f}}_{|\mathcal{C}_r|}\}$ at the transmitter and the receiver sides, where $|\mathcal{C}_t|$ and $|\mathcal{C}_r|$ are the cardinalities of these codebooks correspondingly. In specific, we have $|\mathcal{C}_t| = N_t = |\mathcal{C}_r| = N_r$ in our case. The beam pair $[p, q]$ is converted into a unique index $i \in \{1, 2, \cdots, M\}$, where $M \leq |\mathcal{C}_t||\mathcal{C}_r|$. For the i-th pair, the *effective channel* [39] is calculated as

$$y_i = \mathbf{w}^*_i \mathbf{H} \mathbf{f}_i \qquad (2)$$

and the *optimal* beam pair index $\hat{i}$ is given by

$$\hat{i} = \arg \max_{i \in \{1, \cdots, M\}} |y_i|. \qquad (3)$$

This post-processing and data representations allow the formulation of several ML problems for beam-selection. Some alternatives are discussed in the next section.

IV. Examples of Experiments with 5GMdata

Next we illustrate some ML experiments with the described dataset. Three examples of machine learning problems are described. Only the third example uses the time evolution while the others are *drop-based* and depend only on data from a given scene. In any ML problem, care must be exercised to use an evaluation strategy that allows to estimate the generalization capability of the classifiers. For example, when splitting the dataset into disjoint training, validation and test sets, we shuffle the episodes and not the scenes, given that scenes are similar along an episode. There are many other details in elaborate simulations and, to promote reproducibility, the dataset and associated code will be made available at [43].

A. Conventional drop-based classification

We pose the beam-selection as a classification task in which the target output is the best beam pair index $\hat{i}$. The input features correspond to the matrix described in Section III-C with the following modification: we generate $\mathbf{Q}_{s,r}$, a modified version of $\mathbf{Q}_s$ for each receiver r, assuming a value $+1$ for all $\mathbf{Q}_s$ elements corresponding to the target receiver r, while all other receivers in the given scene s are represented with -1 (instead of their original positive values in $\mathbf{Q}_s$). The 116 episodes (with 50 scenes each) are split and 34 episodes used for testing. For each receiver that is part of a given scene, a classification *example* is obtained, leading to a total of 41,023 examples for training and test. In 16,977 cases, the receiver is in the (larger) RT study area but not in the V2I study area. Among the examples, there is LOS in 25,174 cases and NLOS in 15,849. Transmitter and receivers had 4×4 uniform planar antenna arrays (UPA), such that $N_t = N_r = 16$. There are $M = 61$ classes (optimum beam pairs) among the possible $|\mathcal{C}_t||\mathcal{C}_r| = 256$ pairs. Table II presents the accuracy using this data for distinct learning algorithms. The hyperparameters and other details can be obtained at [43].

TABLE II
Classification results.

	Accuracy (%)	
Classifier	All data	Only NLOS
Linear SVM	33.2	12.4
AdaBoost	55.0	22.5
Decision tree	55.5	27.3
Random Forest	63.2	36.9
Deep neural network	63.8	38.1

The data used for the "All data" column in Table II had approximately 60% of examples in LOS and this is comparable to the maximum accuracy of 63%. The LOS case could be addressed with simple geometry. Restricting attention only to NLOS examples leads to the results in the right column of Table II. In spite of not being our goal to investigate performance levels with this relatively small amount of data, the results indicate that deep learning has clear advantage over the other tested methods. While deep neural networks are more popular, random forest are also "deep" in the sense that consist of an ensemble (obtained with *bagging*) of decision trees [20].

Some classifiers in Table II reach zero errors on the training set while the errors in the test set are relatively large. Such *overfitting* indicates that more data is needed to avoid evaluating deep learning algorithms solely on *small data* regime.

B. Conventional drop-based multivariate regression

The 5GMdata can also be used for regression tasks. For example, estimating the angles of departure and arrival in beam-selection can be cast as a multivariate regression problem in

which the desired output is $[\phi^D_{\ell_*}, \theta^D_{\ell_*}, \phi^A_{\ell_*}, \theta^A_{\ell_*}]$ and the input is the matrix previously used. Table III presents the root mean-squared error (RMSE) for estimations with neural networks in this problem. The deep architecture outperforms the shallow (with only one hidden layer), but in both cases, the deviations from the target are relatively large, especially for the azimuth angles.

TABLE III
REGRESSION RESULTS IN TERMS OF RMSE (ANGLES IN DEGREES).

	Departure (Tx)		Arrival (Rx)	
Regressor	Ele.	Azi.	Ele.	Azi.
Shallow neural network	6.5	137.4	7.9	180.6
Deep neural network	4.8	49.9	6.2	102.8

In some applications, it is not essential to find the optimum beam direction, but grant an overall quality of service. The next subsection presents an example that uses regression to estimate the output powers of each beam within a reinforcement learning setup.

C. Deep reinforcement learning

One of the requirements of our methodology is to provide the history of the channel evolution over time. The time evolution facilitates, for instance, taking into account the interaction of the mmWave PHY with the media access control (MAC) and upper layers. Designing the mmWave MAC is an important issue for 5G [44] and ML can be useful in this context. The next paragraphs aim at giving a concrete example within the framework of *deep reinforcement learning* [20]. Again, the goal is not to outperform previously published methods, but illustrate how 5GMdata can be effectively used.

In reinforcement learning [20], an *agent* is capable of *actions* that influence the *state* of the *environment* to obtain a *reward*. Deep reinforcement learning (DRL) is often associated to having a deep neural network to choose the actions. Fig. 6 indicates how beam-selection can be cast as a DRL problem. In this case, the analog beamforming architecture is assumed and the action is to schedule a user in each time-slot of duration T_{sam}, together with its beam pair index i. The state at scene s is represented by the matrix $\mathbf{Q}_s$ described in Section III-C, which represents all receivers of interest. There is flexibility on choosing the reward. For example, the reward can be based on figures of merit such as throughput, fairness, energy or their combination. Different from conventional regression or classification problems, the agent does not seek to find the optimum solution at each time instant, but properly allocate resources among users to maximize the reward over time.

For simplicity, the rewards are based on the value $z_{s,r,i} = 20\log_{10}|y_{s,r,i}|$ for scene s, receiver r and beam pair i, where $y_{s,r,i}$ is given by Eq. (2). To speed up convergence, the extreme values $z_{\min} = \min_{r,i} z_{s,r,i}$ and $z_{\max} = \max_{r,i} z_{s,r,i}$ for each scene s are used to obtain $\overline{z}_{s,r,i} = (z_{s,r,i} - z_{\min})/(z_{\max} - z_{\min})$ in the range $[0, 1]$. Improved numerical stability is achieved by using a floor value for $z_{\min}$, such as $z_{\max} - 200$. Assuming

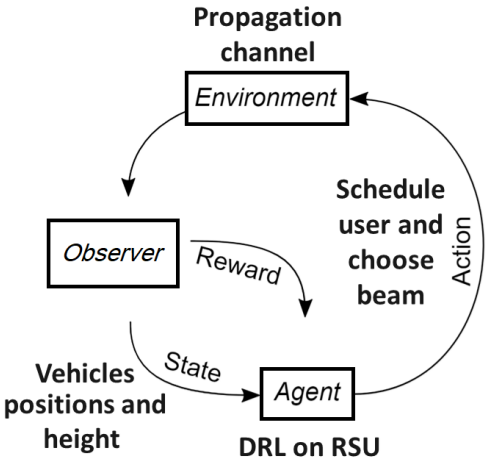

Fig. 6. Deep reinforcement learning for multi-user beam-selection in V2I.

there are N_{sce} scenes per episode, the agent distributes N_{sce} time-slots among the receivers and chooses the beam pair for each one. These decisions are represented by arrays $\mathbf{r}$ and $\mathbf{i}$, whose elements inform the chosen receiver $\mathbf{r}[s]$ and beam pair $\mathbf{i}[s]$ at scene s, respectively. The reward is then $r_s = \overline{z}_{s,\mathbf{r}[s],\mathbf{i}[s]}$ unless there is an *outage*. The user r is in outage when the agent does not allocate time-slots for r over N_{out} or more consecutive scenes. The reward is $r_s = r_{\text{out}}$ if there is any user in outage at scene s. The value of r_{out} is typically a negative number to penalize the outage occurrence. The average reward per scene for an episode e is then $R_e = (1/N_{\text{sce}}) \sum_{s=1}^{N_{\text{sce}}} r_s$. If there is no chance of an outage ($N_{\text{out}} \to \infty$) and assuming an agent capable of always choosing the receiver with the strongest power $z_{s,r,i}$, the average reward would be $R_e = 1, \forall e$, given the way $\overline{z}_{s,r,i}$ is defined.

To model this problem as DRL, we adopt a cascade of two networks. The first is a convolutional deep neural network $\mathcal{N}_1$ that has $\mathbf{Q}_s$ as its input and outputs estimates of $\overline{z}_{s,r,i}$ for scene s, organized as an array with $N_{\text{rec}} \times |\mathcal{C}_t||\mathcal{C}_r|$ elements. The peak value for each row of this array indicate the best beam pair per receiver. This array is part of the input to the second network $\mathcal{N}_2$, which is also composed by an array of binary elements that indicate the time-slots allocated to receivers over the previous N_{out} scenes. The network $\mathcal{N}_2$ has N_{rec} outputs, for each scene s, the receiver that should be allocated to the corresponding time-slot. The output of $\mathcal{N}_1$ is then used to choose the beam pair for the chosen receiver.

Regarding the input to the first network $\mathcal{N}_1$, the matrix $\mathbf{Q}_s$ representing all receivers is converted into N_{rec} matrices $\mathbf{Q}_{s,r}$, that inform where a given receiver r is located, while treating all other receivers as regular vehicles (turning their corresponding positive values in $\mathbf{Q}_s$ into -1), similar to the scheme used in Section IV-A. The input to $\mathcal{N}_1$ is then the concatenation of N_{rec} matrices $\mathbf{Q}_{s,r}$. Conceptually, it would be possible to build for each receiver r_*, a sub-network with input $\mathbf{Q}_{s,r_*}$ that outputs estimates of $\overline{z}_{s,r_*,i}$. To decrease the computational cost, instead, the layers are shared among the

distinct $\mathbf{Q}_{s,r}$ (sharing layers is a feature of DL packages such as Keras [45]). Another speed up is obtained by training $\mathcal{N}_1$ using supervised regression to estimate the outputs $\overline{z}_{s,r_*,i}, \forall i$, given the corresponding inputs $\mathbf{Q}_{s,r_*}$, $r_* = 1, \ldots, N_{\text{rec}}$, A third aspect is that after an action in RL, the environment state needs to be updated accordingly. In our case, it would be inconvenient to execute the RT and traffic simulators within the DRL loop. In this experiment however, invoking the simulators is not necessary given that their pre-calculated outputs suffice to obtain $\overline{z}_{s,r,i}$.

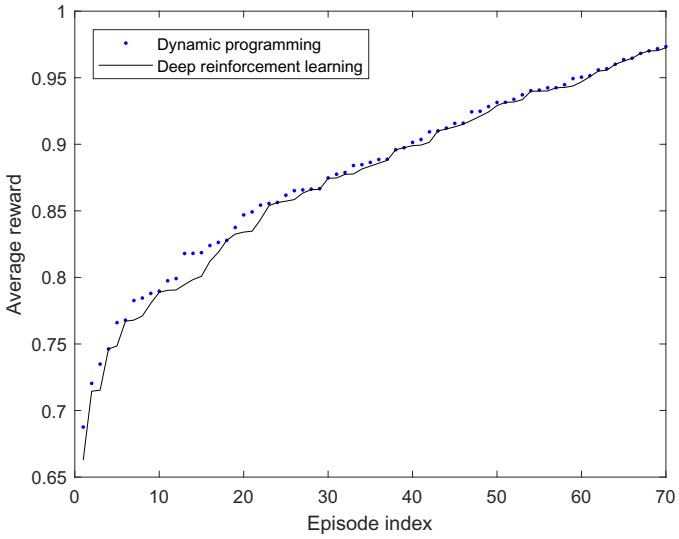

Fig. 7. Performance of DRL in the training set and comparison with the optimum allocation obtained via dynamic programming.

The Deep Q Learning (DQN) algorithm is used, as implemented in Keras-RL [46]. It is assumed $r_{\text{out}} = -3$, $N_{\text{out}} = 3$, $N_{\text{rec}} = 2$ and 4×4 UPAs for both transmitter and receivers. Seventy episodes are used in experiments to illustrate the learning process. The generalization capability is not evaluated in this paper. Considering first the performance of $\mathcal{N}_1$ in the (embedded) supervised regression task, it achieved an average RMSE $= 0.074$ while estimating $\overline{z}_{s,r,i}$. Still in a supervised learning settings, an accuracy of 67.5% is obtained when the strongest beam pair indicated by the estimates of $\overline{z}_{s,r,i}$ is used as a classifier outputs. The performance of the overall DRL model (cascade of $\mathcal{N}_1$ and $\mathcal{N}_2$) is presented in Fig. 7, which shows the average rewards in all episodes. For comparison, assuming the regression values estimated by $\mathcal{N}_1$, Fig. 7 also indicates the optimum time-slot allocation obtained with dynamic programming. The average reward over all episodes for dynamic programming in this case and DRL are 0.879 and 0.874, respectively. If the actual values of $\overline{z}_{s,r,i}$ are passed to the dynamic programming routine (instead of $\mathcal{N}_1$ estimates), the average reward increases to 0.891. These results indicate that the DRL is able to learn the task of simultaneously allocating receivers to time-slots and choosing beam pairs.

V. Conclusion

This paper presented a methodology for generating 5G propagation channel data that decouples the tasks of modeling mobility and channel. This facilitates the use of advanced features of traffic simulators. Given the current lack of freely available large amount of data for benchmarking deep learning algorithms in 5G, it is reasonable to use simulations especially in complicated configurations. The generated data incorporates the channel evolution over time and can be used, for example, in machine learning problems involving aspects of the 5G PHY with constraints from MAC and upper layers. The focus was mmWave MIMO but the methodology can be used in other scenarios. Future work includes simulating different sites and scenarios, while validating some of them with measurements. Currently, it is not clear how detailed must be the description of ray-tracing scenarios to support broad conclusions such as average performance on distinct sites. Measurements can help tuning the methodology. Besides accurate modeling, it is important to minimize the computational cost. An alternative to speed up simulations is to combine ray-tracing outputs with statistical models and eventually avoid the longer simulation time caused by the diffuse-scattering feature. After escaping the small data regime, deep learning in 5G can be investigated using a systematic and reproducible experimental procedure.

Acknowledgment

The work of A. Klautau and P. Batista was supported in part by CNPq, Brazil (processs 201493/2017-9/PDE), and of R. Heath and Y. Wang by the U.S. Department of Transportation through the Data-Supported Transportation Operations and Planning Tier 1 University Transportation Center, in part by the National Science Foundation under Grant 1711702, and by a gift from Huawei. The work of N. González-Prelcic was partially funded by the Agencia Estatal de Investigación (Spain) and the European Regional Development Fund (ERDF) under project MYRADA (TEC2016-75103-C2-2-R).

References

[1] C. Jiang, H. Zhang, Y. Ren, Z. Han, K. C. Chen, and L. Hanzo, "Machine Learning Paradigms for Next-Generation Wireless Networks," *IEEE Wireless Commun.*, vol. 24, no. 2, pp. 98–105, Apr. 2017.

[2] P. V. Klaine, M. A. Imran, O. Onireti, and R. D. Souza, "A Survey of Machine Learning Techniques Applied to Self-Organizing Cellular Networks," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2392–2431, 2017.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[4] T. O'Shea and J. Hoydis, "An Introduction to Deep Learning for the Physical Layer," *IEEE Trans. on Cogn. Commun. Netw.*, vol. 3, no. 4, pp. 563–575, 2017.

[5] T. Y. He, N. Zhao, and H. Yin, "Integrated Networking, Caching and Computing for Connected Vehicles: A Deep Reinforcement Learning Approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 44–55, 2018.

[6] G. Han, L. Xiao, and H. V. Poor, "Two-dimensional anti-jamming communication based on deep reinforcement learning," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Proc. (ICASSP)*, Mar. 2017, pp. 2087–2091.

[7] N. E. West and T. O'Shea, "Deep architectures for modulation recognition," in *Proc. IEEE Int. Symp. Dynamic Spectrum Access Networks (DySPAN)*, Mar. 2017, pp. 1–6.

[8] R. Atallah, C. Assi, and M. Khabbaz, "Deep reinforcement learning-based scheduling for roadside communication networks," in *Proc. Int. Symp. Modeling Optimization Mobile, Ad Hoc, Wireless Networks (WiOpt)*, May 2017, pp. 1–8.

[9] R. F. Atallah, C. M. Assi, and J. Y. Yu, "A Reinforcement Learning Technique for Optimizing Downlink Scheduling in an Energy-Limited Vehicular Network," *IEEE Trans. Veh. Technol.*, vol. 66, no. 6, pp. 4592–4601, Jun. 2017.

[10] Z. Xu, Y. Wang, J. Tang, J. Wang, and M. C. Gursoy, "A deep reinforcement learning based framework for power-efficient resource allocation in cloud RANs," in *Proc. IEEE Int. Conf. Communications (ICC)*, May 2017, pp. 1–6.

[11] P. V. R. Ferreira, R. Paffenroth, A. M. Wyglinski, T. M. Hackett, S. G. Bilén, R. C. Reinhart, and D. J. Mortensen, "Multi-objective reinforcement learning-based deep neural networks for cognitive space communications," in *Proc. Cognitive Communications Aerospace Applications Workshop (CCAA)*, Jun. 2017, pp. 1–8.

[12] "Machine learning for beam based mobility optimization in NR," http://www.diva-portal.org/smash/get/diva2:1088857/FULLTEXT01.pdf, Björn Ekman, Master of Science Thesis, Linköping University, 2017.

[13] J. S. et al, "Natural TTS synthesis by conditioning WaveNet on mel spectrogram predictions," in *https://arxiv.org/abs/1712.05884*, 2018.

[14] G. R. MacCartney, H. Yan, S. Sun, and T. S. Rappaport, "A flexible wideband millimeter-wave channel sounder with local area and NLOS to LOS transition measurements," in *Proc. IEEE Int. Conf. Communications (ICC)*, May 2017, pp. 1–7.

[15] R. W. Heath, N. González-Prelcic, S. Rangan, W. Roh, and A. M. Sayeed, "An Overview of Signal Processing Techniques for Millimeter Wave MIMO Systems," *IEEE J. Sel. Topics Signal Process.*, vol. 10, no. 3, pp. 436–453, Apr. 2016.

[16] T. S. Rappaport, R. W. Heath, R. C. Daniels, and J. N. Murdock, *Millimeter Wave Wireless Communications*. Prentice Hall, 2014.

[17] F. Fuschini, E. M. Vitucci, M. Barbiroli, G. Falciasecca, and V. Degli-Esposti, "Ray tracing propagation modeling for future small-cell and indoor applications: A review of current techniques," *Radio Sci.*, vol. 50, no. 6, p. 2015RS005659, Jun. 2015. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1002/2015RS005659/abstract

[18] M. Rumney, "The critical importance of accurate channel modelling for the success of mmWave 5G," in *Proc. 11th European Conf. Antennas Propagation (EuCAP)*, Mar. 2017, pp. 3688–3691.

[19] J. Turkka, P. Kela, and M. Costa, "On the spatial consistency of stochastic and map-based 5G channel models," in *IEEE Conf. Standards Communications Networking (CSCN)*, Oct. 2016, pp. 1–7.

[20] A. Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2017.

[21] T. Wang, C. Wen, H. Wang, F. Gao, T. Jiang, and S. Jin, "Deep learning for wireless physical layer: Opportunities and challenges," *CoRR*, vol. abs/1710.05312, 2017. [Online]. Available: http://arxiv.org/abs/1710.05312

[22] T. J. O'Shea, T. Erpek, and T. C. Clancy, "Deep learning based MIMO communications," *CoRR*, vol. abs/1707.07980, 2017. [Online]. Available: http://arxiv.org/abs/1707.07980

[23] F. B. Mismar and B. L. Evans, "Partially blind handovers for mmWave new radio aided by sub-6 GHz LTE signaling," in *https://arxiv.org/abs/1710.01879*, 2018.

[24] V. Va, J. Choi, and R. W. Heath, "The Impact of Beamwidth on Temporal Channel Variation in Vehicular Channels and Its Implications," *IEEE Trans. Veh. Technol.*, vol. 66, no. 6, pp. 5014–5029, Jun. 2017.

[25] O. Stabler and R. Hoppe, "MIMO channel capacity computed with 3D ray tracing model," in *3rd European Conf. Antennas Propagation*, Mar. 2009, pp. 2271–2275.

[26] S. Arikawa and Y. Karasawa, "A Simplified MIMO Channel Characteristics Evaluation Scheme Based on Ray Tracing and Its Application to Indoor Radio Systems," *IEEE Antennas Wireless Propag. Lett.*, vol. 13, pp. 1737–1740, 2014.

[27] D. Solomitckii, Q. C. Li, T. Balercia, C. R. C. M. da Silva, S. Talwar, S. Andreev, and Y. Koucheryavy, "Characterizing the Impact of Diffuse Scattering in Urban Millimeter-Wave Deployments," *IEEE Wireless Commun. Lett.*, vol. 5, no. 4, pp. 432–435, Aug. 2016.

[28] "Remcom Wireless InSite," https://www.remcom.com/wireless-insite-em-propagation-software, accessed: 2018-01-20.

[29] Mobile and wireless communications enablers for the Twenty-twenty information society (METIS), "Deliverable D1.4 - METIS channel models (version 3)," EU FP7 Project, Tech. Rep., 2015.

[30] A. F. Molisch, A. Karttunen, S. Hur, J. Park, and J. Zhang, "Spatially consistent pathloss modeling for millimeter-wave channels in urban environments," in *10th European Conf. Antennas Propagation (EuCAP)*, Apr. 2016, pp. 1–5.

[31] Y. Wang, Z. Shi, L. Huang, Z. Yu, and C. Cao, "An Extension of Spatial Channel Model with Spatial Consistency," in *Proc. IEEE 84th Vehicular Technology Conference (VTC-Fall)*, Sep. 2016, pp. 1–5.

[32] "NYUSIM," http://wireless.engineering.nyu.edu/nyusim, accessed: 2018-01-20.

[33] S. Jaeckel, L. Raschkowski, K. Börner, and L. Thiele, "QuaDRiGa: A 3-d multi-cell channel model with time evolution for enabling virtual field trials," *IEEE Trans. Antennas Propag.*, vol. 62, no. 6, pp. 3242–3256, June 2014.

[34] 3GPP, "Spatial consistency modeling in drop based model," 3rd Generation Partnership Project (3GPP), Document for: Discussion and Decision R1-161622, 2016. [Online]. Available: https://portal.3gpp.org/ngppapp/CreateTdoc.aspx?mode=view&contributionId=691847

[35] J. W. McKown and R. L. Hamilton, "Ray tracing as a design tool for radio networks," *IEEE Network*, vol. 5, no. 6, pp. 27–30, Nov. 1991.

[36] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO - Simulation of Urban MObility," *International Journal On Advances in Systems and Measurements*, vol. 5, no. 3&4, pp. 128–138, Dec. 2012.

[37] N. González-Prelcic, A. Ali, V. Va, and R. W. Heath, "Millimeter-wave communication with out-of-band information," *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 140–146, Dec. 2017.

[38] A. Ali, N. González-Prelcic, and R. Heath, "Millimeter Wave Beam-Selection Using Out-of-Band Spatial Information," *IEEE Trans. Wireless Commun.*, vol. 17, no. 2, pp. 1038–1052, 2017.

[39] V. Va, J. Choi, T. Shimizu, G. Bansal, and R. W. Heath, "Inverse Multipath Fingerprinting for Millimeter Wave V2I Beam Alignment," *IEEE Trans. Veh. Technol.*, 2017, IEEE Early access.

[40] J. Kim and A. F. Molisch, "Fast millimeter-wave beam training with receive beamforming," *Journal of Communications and Networks*, vol. 16, no. 5, pp. 512–522, Oct. 2014.

[41] P. Zhou, X. Fang, Y. Fang, Y. Long, R. He, and X. Han, "Enhanced Random Access and Beam Training for Millimeter Wave Wireless Local Networks With High User Density," *IEEE Trans. Wireless Commun.*, vol. 16, no. 12, pp. 7760–7773, Dec. 2017.

[42] S.-C. Kim, B. J. Guarino, T. M. Willis, V. Erceg, S. J. Fortune, R. A. Valenzuela, L. W. Thomas, J. Ling, and J. D. Moore, "Radio propagation measurements and prediction using three-dimensional ray tracing in urban environments at 908 MHz and 1.9 GHz," *IEEE Trans. Veh. Technol.*, vol. 48, no. 3, pp. 931–946, May 1999.

[43] "5GMdata," https://github.com/aldebaro/5gmdata, accessed: 2018-01-20.

[44] S. Dutta, M. Mezzavilla, R. Ford, M. Zhang, S. Rangan, and M. Zorzi, "Frame Structure Design and Analysis for Millimeter Wave Cellular Systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1508–1522, Mar. 2017.

[45] F. Chollet *et al.*, "Keras," https://github.com/keras-team/keras, 2015.

[46] M. Plappert, "Keras-RL," https://github.com/matthiasplappert/keras-rl, 2016.