A Comparative Study of the Computation Efficiency of a GPU-Based Ray Launching Algorithm for UAV-Assisted Wireless Communications

Maximilian J. Arpaio, Enrico M. Vitucci, and Franco Fuschini

Department of Electrical, Electronic and Information Engineering "G. Marconi"
Alma Mater Studiorum University of Bologna, Bologna, 40136, Italy
{maximilian.arpaio, enricomaria.vitucci, franco.fuschini}@unibo.it

Abstract ─ Graphics Processing Units (GPU), have opened up new opportunities for speeding up general-purpose parallel computing applications. In this paper, we present the computation efficiency in terms of time performances of a novel ray launching field prediction algorithm which relies on NVIDIA GPUs and its Compute Unified Device Architecture (CUDA). The software tool assesses the propagation losses between a wireless transmitter - carried by an Unmanned Air Vehicle (UAV) - over a 3D urban environment. Together with other effective features, the software tool is shown to reduce by several orders of magnitude the computation time of simulations. Performances and cost-benefit analysis of three different NVIDIA GPU configurations are thus investigated over three different urban scenarios, taken as test-cases for Air-to-Ground (A2G) communications for 5G applications and beyond.

Index Terms ─ 5G, Air-To-Ground (A2G) propagation, GPU, NVIDIA, ray launching, UAV.

I. INTRODUCTION

Deterministic wave propagation modelling represents a state-of-the-art technique for RF channel analysis. The accuracy of site-specific propagation models, like ray tracing or ray launching, has seen great improvements in the last few decades thanks to better characterization of propagation mechanisms [1]. Although the compute capability of modern processors is constantly increasing, deterministic models still require significant runtimes to achieve accurate results, which has motivated the research community to extensive look for optimized and efficient acceleration methods [2]. Unfortunately, the complex mechanisms of electromagnetic waves as well as the huge amount of geometric calculations can make Central Processing Unit (CPU) computation inefficient. It has been argued that standard models - lacking of any speeding-up expedient - take more than an hour to retrieve channel characteristics across a kilometre-scale scenario with a single radio source [3]. Parallel computing – on the contrary - is a process of decomposing a large serial task into smaller sub-tasks, which can be calculated concurrently. Multi-core processors are currently the most commonly available and exploited parallel computing platform, allowing much faster computations compared to a single core, as long as the computer code is optimized to take advantage from the multiple cores [4]. Furthermore, interest has grown rapidly in recent years towards harnessing the power of graphic hardware to perform general-purpose parallel computing. This alternative approach has become widespread and is based on the use of the GPU—in addition to the CPU—for general purpose computing.

Having effectively reached a limit in the improvement of the single-core frequency of CPUs, GPU computing has become the method of choice for applications with high computational demands. Although GPUs have been known in the computing industry for over 40 years, they haven't represented a breakthrough until programmable and general purpose (GPGPU) have been developed. Since then, the computation potential has gained increasing acknowledgement, and GPUs have become far more than an embedded device for display operations: their special design allows us to perform many operations simultaneously and to perform computation-heavy tasks that would otherwise require a large computer cluster. Moreover, modern GPUs are equipped for double-precision mathematical operations in parallel configuration, which extends the range of applications even further. It is anyway important to note that even if parallelisation is not always guaranteed by simply having installed a GPU card and using CUDA language, it can be potentially achieved at a high- or low-level scale for many applications [5].

By means of a comparative study of the computation efficiency in terms of calculation time, this paper aims at shedding light on the crucial benefits that GPU computing can bring to the characterization of electromagnetic propagation between an UAV flying over an urban area and users roaming at street level. To the best of the authors' knowledge, no papers have been published so far with comparative, detailed, cost-benefit analyses of GPU architectures applied to electromagnetic computation problems. This represents the main novel contribution of

Submitted On: April 26, 2020
Accepted On: October 20, 2020
https://doi.org/10.47037/2020.ACES.J.351201

this paper; surprisingly, high-end, more powerful GPU card might not always provide the greatest computation efficiency as it will be shown in the next sections.

As a matter of fact, while UAV assisted wireless communications are currently envisaged in the framework of 5G and beyond [6], experimental investigations of the A2G link are extremely challenging and complex due to the limited payload, the problem in powering up flying transceivers and the regulations to comply with at national/international level especially within inhabited areas. Therefore, ray-launching simulations can represent an easier and cheaper way to improve awareness about A2G propagation properties, as long as the corresponding computation effort is worth it. In this regard, a key characteristic of the ray launching approach is that rays do not interact each other along their own propagation paths. From departure to arrival, ray paths can be independently traced, which is of great advantage for GPU-based computation thanks to the intrinsic parallelisation degree within the whole propagation process.

This document is structured as follows: in Section II we provide general details of the ray-launching software while in Section III we introduce the hardware and software configurations, with a particular focus on the NVIDIA GPU cards. In Section IV we go through the main results in terms of computation time and speed-up factors, both for isotropic and directive antennas. The final Section V summarises the results while briefly drawing the main conclusions.

II. PRINCIPLES OF RAY-LAUNCHING ACCELERATION

Due to the increase in the computational demands of modern applications, many developers are currently looking for different ways to accelerate their applications beyond the – limited – speed that conventional CPUs can provide. Among all the possible solutions, the baseline for the GPU-based A2G propagation assessment proposed within this paper is the Discrete Environment-Driven Ray Launching model (DED-RL), which has been introduced for the first time in [7] together with the related computational theory. As with all RL algorithms, DED-RL is suitable for prediction over large areas or volumes. More specifically, it has been designed to perform fast deterministic propagation prediction on 3-D outdoor surfaces of all buildings and streets in a given target area, to enable multi frequency RF coverage design and optimization.

The software relies on a digitalised 3D urban model where each building is a polygon prism with a defined shape, material, position and height. The model is totally discrete, i.e., the building walls are properly discretized into "tiles" with a predetermined size. DED-RL has also inherited some advanced features from a pre-existing RT model developed at the University of Bologna, such as the Effective Roughness (ER) diffuse-scattering model. In addition to the environment discretization, the algorithm is also "environment-driven", meaning that ray tubes are launched only towards the tiles that are visible from the transmitter, and these ray tubes are then bounced toward tiles that are visible to each other. Another advantage of the discretization is that all the visibility relations among the tiles can be pre-computed and properly stored into a "visibility matrix" since the tile centres can be assumed as fixed points. This visibility pre-processing takes advantage of GPU parallelization and must be done only one time for a single simulation scenario. Once the pre-processing is done, ray bouncing can be performed very efficiently for any transmitter location in the same environment. All these features are implemented in DED-RL through the CUDA C++ language for NVIDIA GPUs. Using the combination of the above-mentioned techniques in addition to GPU parallelization, DED-RL is thus able to achieve very high levels of computational efficiency – up to four orders of magnitude compared to a conventional ray-tracing algorithm – while retaining a good level of accuracy, despite the intrinsic error introduced by the environment discretization [8].

All the main features of DED-RL algorithm described above — visibility pre-processing, launching of ray tubes, ray bouncing, and field computation — are suitable for code parallelization via GPU acceleration and thus may benefit significantly from GPU-based computation due to its ability to process vectors or matrices with extreme efficiency, as it will be shown in the following sections.

III. HARDWARE CONFIGURATION AND SIMULATION MODELLING

In order to investigate the computation time for UAV A2G propagation, we performed worst-case ray-launching simulations of a full three-dimensional (3D) scenario. DED-RL simulations were run by means of dedicated scripts within a MATLAB R2017B (Update 9) environment. The purpose of the scripts was to automate in a simple and efficient way the different runs concerning UAV positions and flight levels, as well as its transmitting frequencies.

Table 1: NVIDIA GPUs configurations under investigation

GPU Card	Architecture	Streaming Processors	Core Clock	Memory Clock	Bus Width	VRAM	Single Precision	Double Precision
Tesla K40c	Kepler, GK180	2880	745 MHz	6 GHz GDDR5	384-bit	12GB	5.04 TFLOPS	1.68 TFLOPS
Titan XP	Pascal, GP102	3840	1405 MHz	11.4 GHz DDR5X	384-bit	12GB	12.15 TFLOPS	0.38 TFLOPS
Tesla P100	Pascal, GP100	3584	1190 MHz	1.4 GHz HBM2	4096-bit	12GB	9.32 TFLOPS	4.73 TFLOPS

To freeze the hardware configuration throughout the different runs, all simulations were set-up on a commercial workstation, equipped with an Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10 GHz [8c/16t] and 48 GB DDR4 RAM.

As listed and described in Table 1, three different NVIDIA GPU cards were set-up: two cards belonging to the professional business sector (Tesla series), namely Tesla K40c (medium end) and Tesla P100 (high end), while the last one belonging to the gaming business one (GTX series), namely Titan Xp (high end). Although an extensive description of hardware details and specific mechanism of NVIDIA GPUs is out of the scope of this paper, the reader can find interesting details in [9] for Kepler and in [10] for Pascal architectures.

Regardless of the specific business sector they have been designed for, GPU processing capabilities can be measured in terms of SP and Video RAM (VRAM), together with floating point operations per second, either single or double precision. It can be seen from Table 1 that the three NVIDIA GPU cards show the same VRAM but they differentiate from each other for specific features. The Tesla series cards, as expected for professional business purposes, show better performances in terms of double precision TFlops, while the GTX card, gaming-oriented, really lacks. Conversely, the GTX card outperforms the Tesla series concerning single precision TFlops and memory clock, as expected from a card that must react promptly in tough gaming sessions. The number of SP is comparable between the two high-end GTX and Tesla cards, being instead slightly lower in the medium-end Tesla card. It should be remarked that performance improvements are an increasing function of the number of available computing cores; the more cores are available, the higher the speedups that can be achieved compared to sequential counterpart versions.

Together with the complexity of device architectures, it is seen that computational power of GPUs is rapidly growing with many new features proposed to developers, to the researchers or to the gaming community, like the very recent Ray Tracing (RT) cores for real time ray-tracing calculations [11]. Nevertheless, we must not be inebriated by the multiple features and capabilities: one of the most important aspects when comparing performances in terms of computation times, is to get a fair and balanced set of output metrics for proper accelerator comparison. In this regards Table 2 summarises the main DED-RL parameters set-up during the different simulation runs. In fact, it was important to fine-tune the DED-RL parameters - among those related to the addressed GPU memory and the number of rays launched per cycle - with a set of commonly acceptable values for all the involved GPUs and to get comparable results among the different scenarios.

For benchmarking purposes, the DED-RL software was configured assuming a single UAV hovering over a 3D urban city environment at different positions in space.

Table 2: Ray launching main simulation parameters

Parameter	Values
Frequency	0.7, 3.5, 26 and 70 GHz
UAV heights	30, 50, 75, 150, 300, 450 m AGL
UAV hovering positions	8 circular positions
Number of Interactions	5 bounces, 5 reflections, 2 diffractions and 1 scatter
Number of Combined Interactions	3 reflections/diffractions (max), 3 diffractions/scatters (max)
GPU Memory Allocation Heap Size	1536 MB
Maximum LOS Rays Per Cycle	100000
Amount of GPU Memory for Packets	40%

Table 3: Urban model environments

Parameter	Bologna	Munich	San Francisco
No. Tiles	170931	148584	268868
Area [Km^2]	6.5	8.8	10.2
Building/Area	30.5%	37.8%	40.5%

Three different urban models were investigated, with a special focus on their city centres: Bologna (Italy), Munich (Germany) and San Francisco (USA). This was done to explore how much a specific urban map was affecting the computation time. Coverage predictions were performed on a whole urban area with a single tile resolution of 10x10 m and general details as further specified in Table 3.

By means of *tic* and *toc* Matlab commands [12], it

was possible to measure - and to focus only on - the elapsed time before and after the call to the executable DED-RL file.

Although, on one hand, this is the most straightforward way to measure the computation time in a coherent way among the different GPU cards, on the other hand it may be objected that the CPU processing affects this measurement as part of the whole RL code execution. We can say not only that this part is negligible, but we must also emphasise that this part is common – and the same – for any simulation run on the same workstation, thus returning a set of comparable results.

IV. RESULTS

Simulation results shown in the following subsections represent the combined outcomes of multiple aggregated runs corresponding to different UAV spatial hovering positions (i.e., lat-long UTM coordinates and height above ground level,) or transmitting frequencies, over the three different test-referenced scenarios. This strategy was agreed to make available significant data samples and a clear breakdown of run parameters and characteristics.

A. Computation time for isotropic antennas

In this specific subsection, the UAV was equipped with an isotropic antenna. Although this type of antenna does not have any physical meaning, it allowed us to run a worst-case scenario where rays were launched in all directions, thus increasing the computation effort of calculating multiple interactions at 360° spherical degrees. Furthermore, it is worth noting that the isotropic case can somehow represent real situations where the radiation lobe of the antenna is wide enough to illuminate the whole urban area below the UAV frame.

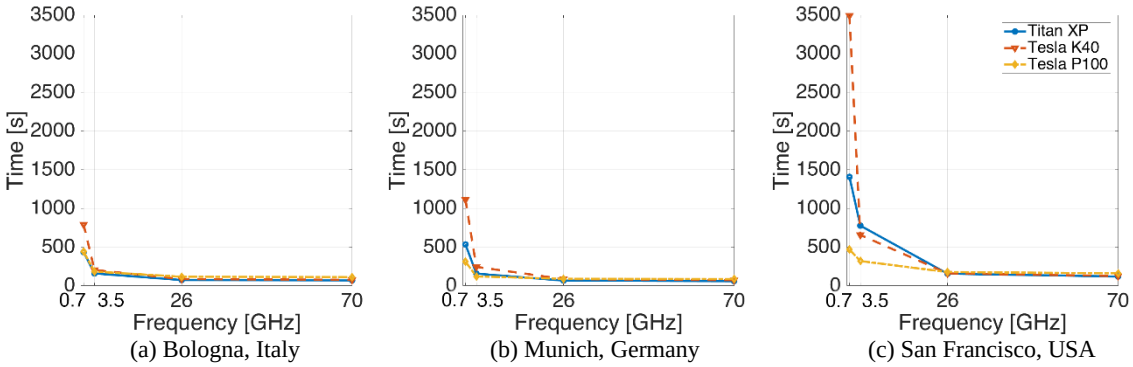
To get the representative graphs in Fig. 1, runs have been averaged over the UAV spatial positions, as it turned out to slightly affect the computation time. According to Fig. 1, this time is generally longer at lower frequencies and shorter at higher frequencies. Differently from "Image" Ray Tracing techniques, where the intensity of a ray can be computed only after the whole ray path - from the transmitter to the receiver - is traced, Ray Launching can take note instead of the ray intensity while it is being traced. Therefore, rays with negligible intensity can be stopped and discarded, thus saving computation time. As propagation losses increase with frequency and distance, many rays are therefore dismissed by the DED-RL algorithm, thus explaining the achieved results.

At the same time, the Tesla P100 GPU card tends to be the fastest one at lower frequencies, with an average speed up factor of ~4x vs. Tesla K40c and ~2x vs. Titan Xp. This speed-up factor is seen to increase with the complexity of the environment. In the most challenging case (San Francisco, according to the parameters listed in Table 3), the Tesla P100 shows a simulation time which is ~3x and ~7x lower compared to the Titan Xp and the Tesla K40 cards respectively, whereas the three GPU cards show similar performances from 26 GHz onwards. On one hand, these results are reasonable and proportional to the number of rays vs. computed interactions. On the other hand, this sounds like a surprising result: it might be expected the high end Tesla P100 card to always be somehow the first of the class due to its technical specs and economical value, while actually the gap with the Titan Xp is indeed minor.

In agreement with CUDA programming best practices [13] and literature [14], this behaviour is likely related to the additional overhead the Tesla P100 brings in connection to its intrinsic complexity. This can limit the computation speed when there is no good balance among the different thread blocks scheduled onto the GPU Streaming Multiprocessors (SM). However, this imbalance is more likely to happen in less challenging cases, i.e. the higher frequencies, due to the selective discarding of those rays whose intensity falls below the minimum power threshold, as previously mentioned.

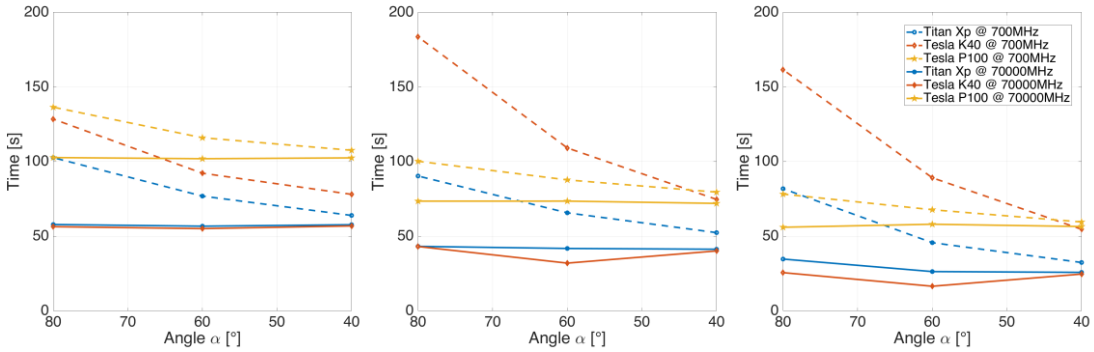
B. Computation time for directive antennas

Following the interesting results of the previous subsection, the UAV was then equipped with a directional antenna of fixed aperture angle α, placed under the UAV fuselage and pointing downwards.



(a) Bologna, Italy (b) Munich, Germany (c) San Francisco, USA

Fig. 1. Computation time [s] over frequency [GHz] for different Nvidia cards and environments.



(a) Bologna, Italy (b) Munich, Germany (c) San Francisco, USA

Fig. 2. Computation time [s] over antenna aperture angle [°] for different NVIDIA cards and environments.

This can be seen as a best-case scenario where rays are launched only within a specific cone, thus drastically reducing the computation effort of calculating additional interactions. In this regard, please note that angle α is simply the ideal radiation cone aperture, i.e., we assumed a simplified 0 dB constant gain inside and instead an –Inf dB constant gain outside.

To get the representative graphs in Fig. 2, runs have been averaged in terms of spatial positions and results split into low and high frequency samples, 700 MHz and 70GHz respectively. On that note, it is seen that at 70GHz the simulation time are flat all over the α angle span, with the Tesla P100 now the slowest among the card.

As the limited number of rays to be traced at high frequency is further reduced by the antenna directive pattern, the simulation time is simply dominated by the specific overhead of the GPU, which is likely to be heavier for the Tesla P100. This is not completely true at 700MHz, where plots are no longer flat and simulation time logically decreases as a function of the angle α, (i.e., it increases as a function of antenna directivity).

Generally speaking, the use of directive antennas in our test-case brings out the way in which GPU overhead represents an important factor for any evaluation of the computation time. From these plots, it is possible to see the Titan XP card to better perform out of the other two cards, which is not always true in case of an isotropic antenna.

V. CONCLUSION

We have investigated the performance of a Discrete Environment-Driven Ray Launching Algorithm in terms of computation time and the related speed-ups among different NVIDIA GPU cards.

We have demonstrated the benefit of GPU parallelization as a means to accelerate ray launching field computation, with typical computation times for complete predictions over all building surfaces ranging from seconds to few tens of minutes, depending on the size of the urban scenario, the hardware used for simulation runs and the characteristics of RF propagation. This shows the potential benefit of GPUs for electromagnetic simulations, and especially for deterministic field strength predictions, in fair agreement with the main outcomes of previous works in [15-17].

It was seen how computation time decreases with frequencies and the use of different directive antennas could affect simulation time. As it can be expected, the wider the antenna radiation cone, the longer the simulation time, although remarkable only at lower frequencies. It was also seen that both professional and gaming GPGPU provide reasonable and consistent results in terms of computation time, the former having better performances at lower frequencies due to the higher number of rays to be processed. On the other hand, performance can be degraded when using the high-end Tesla GPUs in less demanding environments, due to their additional overhead. This shows that NVIDIA gaming GTX cards should not be automatically dismissed and they can be a good choice under specific simulation cases, instead of more expensive Tesla cards.

Future works will focus on more demanding simulation environments, i.e., densely urban as well as on different and more recent Nvidia GPU architectures, especially those with RT cores, like Turing and Ampere.

ACKNOWLEDGMENT

The authors wish to express their gratitude to NVIDIA Corporation with the donation of the Titan XP GPU used for this research.

REFERENCES

[1] Z. Yun, M. F. Iskander, “Ray tracing for radio propagation modelling: Principles and applications,” *IEEE Access*, vol. 3, pp. 1089-1100, July 2015.

[2] N. Kinayman, “Parallel programming with GPUs: Parallel programming using graphics processing units with numerical examples for microwave engineering,” *IEEE Microwave Magazine*, vol. 14,

iss. 4, pp. 102-115, June 2013.

[3] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Kruger, A. E. Lefohn, and T. J. Purcell, "A survey of general-purpose computation on graphics hardware," *Proc. Eur. Assoc. Comput. Graph*., pp. 21-51, Aug. 2005.

[4] A. Hidic, D. Zubanovic, A. Hajdarevic, A. Huseinovic, and N. Nosovic, "Attempt of unbiased comparison of GPU and CPU performance in common scientific computing," *2012 IX International Symposium on Telecommunications (BIHTEL)*, Sarajevo, Bosnia & Herzegovina, Oct. 25-27, 2012.

[5] S. W. Keckler, W. J. Dally, B. Khailany, M. Garland, and D. Glasco, "GPUs and the future of parallel computing," *IEEE Micro*., vol. 31, iss. 5, Sept.-Oct. 2011.

[6] US Dept. of Transportation. "Unmanned Aircraft System (UAS) Service Demand 2015-2035: Literature Review and Projections of Future Usage," Technical Report, v. 1.0, Feb. 2014.

[7] J. S. Lu, E. M. Vitucci, V. Degli-Esposti, F. Fuschini, M. Barbiroli, J. A. Blaha, and H. L. Bertoni, "A discrete environment-driven GPU-based ray launching algorithm," *IEEE Transactions on Antennas and Propagation*, vol. 67, iss. 2, pp. 1180-1192, Feb. 2019.

[8] E. M. Vitucci, V. Degli-Esposti, F. Fuschini, J. S. Lu, M. Barbiroli, J. N. Wu, M. Zoli, J. J. Zhu, and H. L. Bertoni, "Ray tracing RF field prediction: An unforgiving validation," *International Journal of Antennas and Propagation*, Hindawi, vol. 2015, pp. 1-11, Aug. 2015.

[9] NVIDIA Corporation Whitepaper, "NVIDIA's Next Generation, CUDA Compute Architecture: Kepler GK110/210 Family," 2012.

[10] NVIDIA Corporation Whitepaper, "NVIDIA's Next Generation, CUDA Compute Architecture: Pascal GP100 Family," 2017.

[11] NVIDIA Corporation Whitepaper, "NVIDIA's NVIDIA Turing GPU Architecture: Graphics Reinvented," 2019.

[12] The MathWorks, Inc., "Matlab 2017B User Guide: On-line Help," referenced resources, 2016.

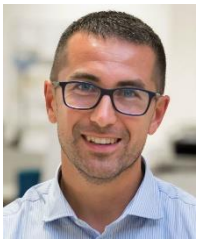
[13] NVIDIA Corporation, Developer's zone, "NVIDIA CUDA Toolkit 10.2.89: CUDA Toolkit Documentation," Nov. 2019.

[14] N. Matloff, "Parallel Computing for Data Science: With Examples in R, C++ and CUDA," June 4, 2015 Chapman and Hall/CRC Pub., June 2015.

[15] C. Reaño and F. Silla, "Performance evaluation of the NVIDIA Pascal GPU architecture: Early experiences," *2016 IEEE 18th International Conference on High Performance Computing and Communications*, Sydney, Australia, Dec. 12-14, 2016.

[16] Z. Dai and R. J. Watson, "Accelerating a ray launching model using GPU with CUDA," *12th European Conference on Antennas and Propagation (EuCAP 2018)*, London, UK, Apr. 9-13, 2018.

[17] M. Ujaldon, "Using GPUs for accelerating electromagnetic simulations," *Applied Computational Electromagnetics Society Journal*, vol. 25, no. 4, 2010.



Maximilian James Arpaio received the Master degree in Telecommunications Engineering from the University of Parma in 2005 and a specialization in wind engineering and aerodynamics from the Polytechnic University of Milan in 2007. He received a Postgraduate Master in Project Management from the University of Verona in 2012. Since 2006, he has been collaborating with various Italian universities by promoting technical seminars and scientific collaborations. His current interests are on antennas and RF propagation within different environments, especially those related to next generation mobile systems (5G) for UAVs assisted wireless networks. He is a member of the IEEE and the Antennas & Propagation Society since 2018.

Enrico Maria Vitucci received the M.Sc. degree in Telecommunication Engineering and the Ph.D. degree in Electrical Engineering from the University of Bologna, Italy. He is currently a tenure-track Assistant Professor in electromagnetic fields at the Department of Electrical, Electronic and Information Engineering "G. Marconi" (DEI) of the University of Bologna. From 2011 to 2016, he was a Research Associate at the Center for Industrial Research on ICT of the University of Bologna. In 2015, he was a Visiting Researcher at Polaris Wireless, Inc., Mountain View, USA. His research interests are in deterministic wireless propagation models and multi-dimensional radio channel characterization. He is author or co-author of about 80 technical papers on international journals and conferences, and co-inventor of 4 international patents. He is a Senior Member of IEEE, and a member of the Editorial Board of the journal "Wireless Communications and Mobile Computing".

Franco Fuschini graduated with honours in Telecommunication Engineering and received the Ph.D. degree in Electronics and Computer Science from the University of Bologna in March 1999 and in July 2003, respectively. In April 1999 he received the 'Marconi Foundation Young Scientist Prize' in the context of the XXV Marconi International Fellowship Award. He is now Research Associate at the Department of Electrical, Electronic and Information Engineering "G. Marconi" of the University of Bologna. His main research interests are in the area of radio systems design and radio propagation channel theoretical modelling and experimental investigation. Franco Fuschini is author or co-author of about 20 papers on IEEE journals about radio propagation and wireless system design.