INTERNATIONAL JOURNAL OF COMMUNICATIONS
Issue 4, Volume 6, 2012

Shooting and Bouncing Ray Approach for 4G Radio Network Planning

Andrej Vilhar, Andrej Hrovat, Igor Ozimek and Tomaž Javornik

***Abstract*—**Radio signal propagation prediction plays an essential role in planning of modern broadband mobile radio networks. While the less accurate statistical models predict only the path loss, the more precise ray tracing techniques enable computation of additional parameters, e.g. the mean delay and the delay spread, which are significant for the 4G network planning. The existing ray tracing methods are mainly applied in the indoor and narrow urban environments. In this paper, an open-source solution for propagation prediction with a ray tracing method in the open urban and rural areas is presented. The emphasis is on the implementation of a computationally efficient procedure, which is divided into two separately optimized problems, tessellation and ray tracing. Ray tracing is implemented by applying an improved shooting and bouncing ray (SBR) approach. The efficiency of the implementation is analysed in terms of computational complexity and results accuracy. The simulation results show that by applying a tessellation optimization technique the processing time is reduced up to 50%. The improved SBR ray tracing approach has been incorporated in the existing open source radio planning tool GRASS-RaPlaT.

***Keywords*—**GRASS-RaPlaT, path loss, propagation prediction, radio planning, ray tracing, SBR, tessellation.

I. Introduction

REQUIREMENTS of emerging applications and user behaviour steadily demand higher data rates, more reliable services and broader signal coverage. To meet these demands, radio networks have grown in complexity. Therefore, an efficient and accurate radio planning of such systems is required. Radio planners have to maximize the geographical size of the signal coverage, maximize the bandwidth efficiency, minimize interferences and minimize the overall system cost providing a sufficient network capacity. Radio planning of mobile communication networks in the rural environment, including the third generation of mobile communication systems, is based on path-loss radio channel models originated from the empirical Okumura-Hata model [1]. In the urban and suburban environments, combinatorial models based on the recursive method for street microcells [2] and the half screen approach proposed in [3] are frequently used. By exploring deterministic radio wave propagation mechanisms such as diffraction, reflection, absorption and refraction, further improvements in the accuracy of the radio network planning can be achieved. Empirical-deterministic models are applied in several commercial software tools for signal coverage prediction.

An extensive number of various commercial tools for signal coverage and network planning are available on the market. Their list and short surveys can be found in [7]. Examples of more sophisticated and thus more expensive and not widely spread tools, which are designed mainly for mobile operators and national regulators, are Planet, decibel Planner, Vulcano and CS telecom nG. Cheaper but functionally limited tools often intended for a specific narrow task are also available such are WinProp, Wireless InSite or TAP, as well as some custom built technology-specific tools developed by hardware development companies [8]. These tools do not comprise modules for radio network optimization and are intended for specific tasks such as WLAN network planning, calculation of radio coverage inside buildings, design of radio-relay links, etc. Commercial radio coverage tools in general do not allow their users to access or modify implemented functionalities, which makes them of limited use for research and development. These limitations can be avoided by using an open-source platform which can be upgraded by an arbitrary propagation model. An example of an existing open-source radio planning tool is Q-Rap, which has been released as a plug-in for the Quantum GIS (QGIS) open source system [9]. Its propagation prediction models are based on the free-space loss calculations, improved by accounting for the knife-edge diffractions and losses due to rounded hills.

While the path loss prediction is sufficient for radio planning of narrowband communication systems, the delay spread and mean delay play an important role in designing of broadband wireless networks in order to correctly tune the parameters of a communication system, for example the cyclic prefix in the orthogonal frequency digital multiple access (OFDMA).

OFDMA is the technology standardized in the 4G and beyond communication systems [4]. A couple of years ago, OFDMA was foreseen as a technology for systems operating indoors and in highly populated urban environments. However, by switching off the analogue television and introducing the digital TV, a part of electromagnetic spectrum has been freed up and this spectrum is/will become available for mobile applications. Due to the radio wave propagation characteristics, the spectrum previously allocated for the television broadcasters is perfect for the provision of wireless

A. Vilhar, A. Hrovat, I. Ozimek and T. Javornik are with the Department of Communication Systems, Jozef Stefan Institute, Ljubljana, Slovenia (phone: +386-1-477-3132; fax: +386-1-477-3111; e-mail: andrej.vilhar@ijs.si).

access in the remote rural areas. According to the network providers, LTE and LTE advanced, both based on the OFDMA multiple access, are foreseen as the key technologies in all geographical areas. These trends in the deployment of the next generation of communication systems require the introduction of new approaches for the radio signal coverage calculation.

Deterministic approaches based on the ray tracing techniques estimate the path loss with an outstanding accuracy. At a particular receiver position, the time-space impulse response, which is an important parameter in wireless broadband system design, can be determined. Since the ray tracing techniques assess the actual distance of radio waves between the transmitter and the receiver considering direct, reflected and diffracted rays, the delay spread and mean delay can be determined.

The rapid development of computer capabilities make the ray tracing techniques an interesting approach for radio signal coverage calculations in various environments. These techniques are particularly attractive because of their accuracy in predicting the path loss, delay spread and mean delay. At first the ray tracing approach was applied for the dense urban areas and in various indoor environments [5]. Lately, some basic implementations of the ray tracing elements for the hilly rural environments have been introduced [6]. However, the computational time needed for the 3D ray tracing calculation of radio coverage on the open areas is still too long.

Ray tracing has been deeply studied in the field of computer graphics, and many algorithms have been published. However, those findings are not directly applicable for use in radio propagation tools, and the algorithms have to be adjusted. In this paper, we propose innovative approaches which will speed up ray tracing applied for the radio coverage prediction in rural and urban areas, and prepare the relief data for the input into the ray tracing engine.

Our solution, GRASS-RaPlaT, is built around the open-source Geographical Resources Analysis Support System (GRASS). In addition to various empirical and combinatorial propagation models, the radio coverage tool also includes modules incorporating innovative approaches for fast 3D ray tracing in the rural and urban environments, which are described in the subsequent sections of the paper.

The paper is organized as follows. The second section briefly introduces the GRASS-RaPlaT tool including the ray-tracing approach. Next, a description of the advanced tessellation algorithms is given. The ray tracing algorithms and their efficient implementation are explained in section 4. In the following section, the simulation results are discussed. The paper concludes with remarks about further improvements of the algorithm and pros and cons of the implemented algorithm in the GRASS-RaPlaT software tool.

II. Ray-Tracing in GRASS-RaPlaT

The innovative ray-tracing procedure, depicted in Fig.1, is implemented as a part of an open source radio propagation tool called GRASS-RaPlaT ("RAdio PLAnning Tool"), introduced in [10] and published online [11]. GRASS-RaPlaT is an add-on to the GRASS open source geographical information system (GIS) tool [12]. It is composed of two basic groups of modules. The core part comprises GRASS modules for radio coverage calculations, which can be linked together with a script written in the Python programming language. Additional modules for data comparison and for adapting input data to the GRASS data structure constitute the second group of GRASS modules. The core of the radio coverage software is radio coverage calculation for the whole network which is divided into three steps:

- path loss calculation for an isotropic source,
- calculation of radio coverage for a particular antenna type and installation (position, direction) and
- generation of a complete coverage data for the whole radio cell network.

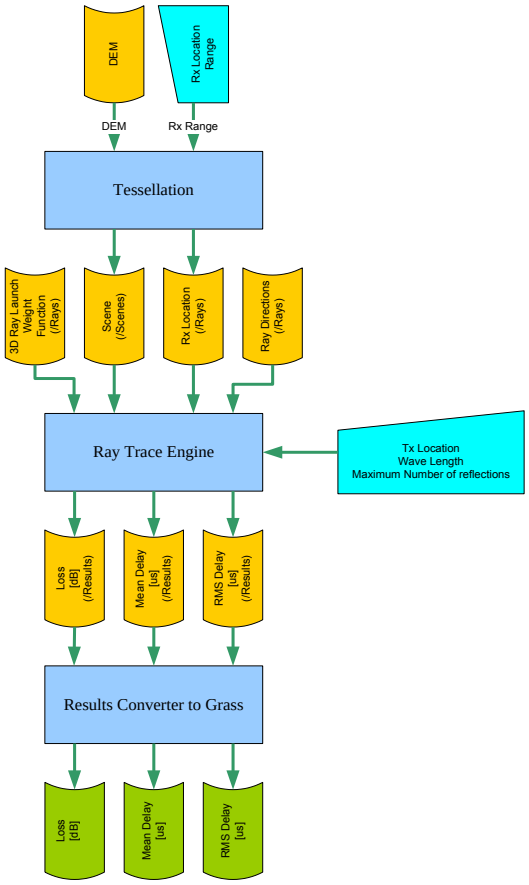

Fig. 1: The program architecture for path loss calculations by using ray-tracing methods

The whole procedure is automated by a Python script, *r.radcov*, which reads the radio network configuration data in tabular form from an input file and calls individual radio coverage computation modules as required. The modularity of GRASS-RaPlaT enables simple upgrading or substitution of the existing mathematical models, module independency from a specific radio network, and quick and simple recalculation

for an individual segment or chosen geographical region.

In the previous version of GRASS-RaPlaT, statistical and combinatorial models have been used for the first step. In the improved version, ray-tracing principles have been included. While the statistical and combinatorial models, which are based on mathematical equations, are relatively simple to implement, a method incorporating a ray-tracing technique requires a more elaborate approach [13].

The GRASS-RaPlaT ray-tracing modules are subdivided into three subgroups (Fig. 1), typically run successively:

- subgroup for tessellation,
- subgroup for ray-tracing,
- subgroup for data conversion.

The common method to represent relief in rural areas is using the raster format, known as the digital elevation model (DEM). The raster data format is an inappropriate data structure for any ray tracing algorithm. Thus the first step is to transform a relief represented as DEM data into the vector form. The common name for this procedure is tessellation. Tessellation actually transforms the relief raster data into a set of triangles. The calculation time of the algorithm directly depends on the number of triangles in the relief representation. In the paper, efficient algorithms are proposed which find a minimum number of triangles for a sufficiently accurate representation of both the rural and the urban areas. The set of triangles is the input to the ray tracer. The second step is designing an efficient ray tracer, which includes the selection of the ray tracer algorithm itself and its input parameters to obtain accurate results in an arbitrary area. We discuss the approaches known from the ray tracing in computer graphics and select the most appropriate one for the implementation in the radio ray tracer.

III. TESSELLATION

In the first subgroup, the input digital relief model in the raster format is converted into the vector (mosaic) format. We chose the triangle as the basic element, which can be used to represent any type of terrain with a selected accuracy. To perform triangulation, we implemented three modules, *r.triangles*, *r.refine* and *r.UrbanTriangles*. The *r.triangles* module is a basic module, developed from scratch and universal in use. The *r.refine* module was developed on the grounds of another open-source project [14]. Its performance is optimal in the rural environment. The module *r.UrbanTriangles* has also been developed from scratch and is especially suited for the tessellation of the urban environment. The description of all three modules is given bellow. The tessellation process concludes by distributing ray receivers over the inspected terrain by another module called *r.PlaceReceivers*.

A. r.triangles Algorithm

The *r.triangles* module finds triangles by simply interconnecting neighboring triplets of points. An example output is depicted in Fig. 2 in the x-y plane, and in Fig. 3 as the corresponding 3-D plot. All the triangle vertices represent individual points of the digital elevation model (DEM) and vice versa.

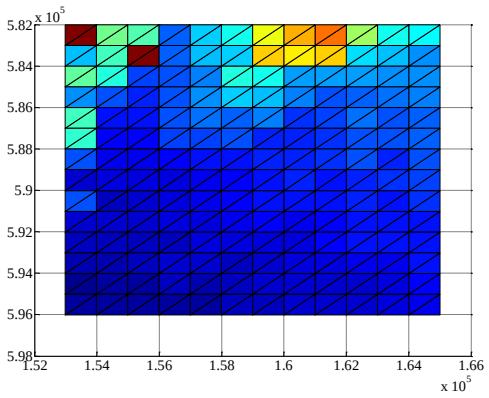

Fig. 2: An example of triangulation performed by *r.triangles* in x-y plane

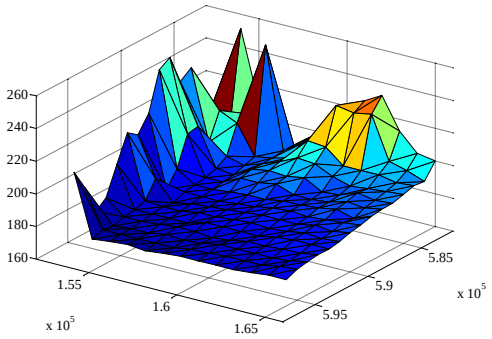

Fig. 3: An example of triangulation performed by *r.triangles* in 3-D plot

The number of points and hence the number of triangles can be adjusted prior to the module execution by setting the desired resolution in the GRASS environment. The main advantages of the module are its simple implementation and usage. The main disadvantage is its suboptimal terrain representation, requiring more triangles for the desired accuracy.

B. r.refine Algorithm

The *r.refine* algorithm optimizes triangle searching based on the terrain dynamics, where flat regions require fewer triangles, while hilly regions are represented by more elements. By this method, terrain representation becomes more efficient, meaning that either the same number of triangles provides higher accuracy or for the same selected accuracy fewer triangles are required.

The algorithm searches for vertices among the existing DEM points following the Heller heuristics [15]. The procedure starts with four corner points in the given raster, which define two triangles. Then, points are added iteratively such that the error ε, depicted in Fig. 4 becomes smaller than a selected value. Besides tuning the ε, the accuracy of

triangulation can be regulated also by setting the resolution in GRASS, i.e. the number of input DEM points.

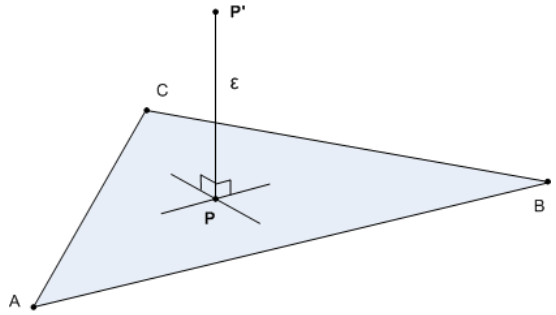

Fig. 4: An error ε is the largest distance between the triangle and the original raster

The obtained points are finally interconnected by running the well-known Delaunay triangulation [16], which creates so-called "fat" triangles with maximized minimal angles.

Fig. 5 and Fig. 6 depict an example output of *r.refine* in the x-y plane and as a 3-D plot, respectively.

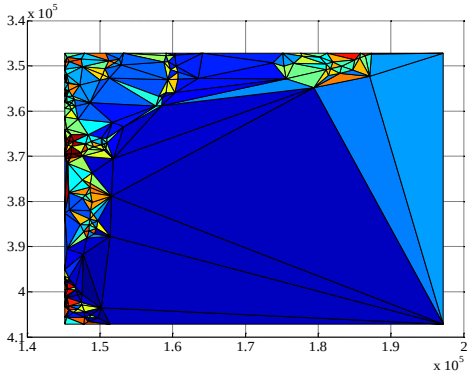

Fig. 5: An example of triangulation performed by *r.refine* in x-y plane

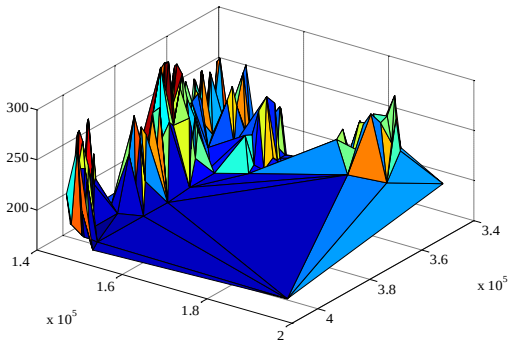

Fig. 6: An example of triangulation performed by *r.refine* in 3-D plot

Compared to *r.triangle*, only some of the input DEM points are selected for vertices. Comparison of Fig. 3 and Fig. 6 reveals that with a similar number of output triangles (336 and 348, respectively), the accuracy in the flat regions is comparable, while the height peaks are much better represented in the latter case.

C. *r.UrbanTriangles Algorithm*

In an urban environment, the terrain is typically flat, while the buildings have sharp edges. In such an environment none of the above modules perform well. Another module, *r.UrbanTriangles*, had to be developed for this purpose.

It is based on the idea that DEM points in flat regions have to be recognized and eliminated before the triangulation, as they do not contain useful information. The algorithm keeps the points on the edges of buildings, such that their shape is retained, while the points in the middle of flat areas (on the ground, roofs, walls, etc.) are eliminated. With this procedure, the triangles in the flat areas are enlarged, while the edges remain sharp.

After the elimination, the remaining points are interconnected by using the publicly available Triangle algorithm [17]. The final number of triangles can be regulated by predefining the resolution in GRASS.

In Fig. 7, a comparison of the three presented algorithms is illustrated. The following numbers of triangles were required to present the analysed area: 498002 by *r.triangles*, 112377 by *r.refine* and 104187 by *r.UrbanTriangles*. The comparison reveals that the *r.triangles* algorithm may be very accurate, but at the cost of wasteful representation, while the *r.refine* returns the opposite results, i.e. an efficient representation at the cost of inaccuracy. The module *r.UrbanTriangles* combines the advantages of the previous modules by a relatively efficient and accurate representation.

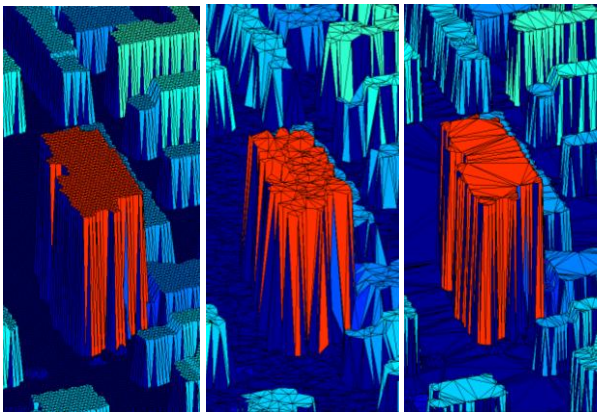

Fig. 7: Comparison of tessellation with *r.triangles* (left), *r.refine* (middle) and *r.UrbanTriangles* (right)

IV. Ray-Tracing Approach for Modelling Radio Propagation

Ray tracing is a technique widely applied in computer graphics for picture generating and rendering. It also finds applications in the radar techniques and wireless communication systems for radio channel modeling. Radio wave propagation can be represented as a set of rays traveling from a transmitter to a receiver. The rays are reflected,

absorbed and diffracted from facets, and due to this their path lengths vary. Consequently, they hit the receiver with various phases, which results in signal fading and channel delay spread. Representing the radio wave propagation with radio rays is known as the geometrical optics (GO) approach. It can be applied for radio channel modeling only if the facets representing the propagation environment are large compared to the signal wavelength. This condition is not too severe for the contemporary telecommunication systems operating in the centimeter frequency bands. The main drawback of the ray tracing approach is that it requires a detailed description of the propagation environment. If this description is too precise, the number of facets is too high and the computation time will exceed any reasonable value. The second problem is that the electromagnetic characteristics of a single facet is usually not known, which results in erroneous amplitude and phase values of the reflected rays. These are the two main reasons that the GO approach is applied mostly in well-defined propagation environments with low number of facets, such as inside buildings and in parts of city centers.

The basic task of ray tracing is to find radio path from a transmitter to a receiver. There exist two basic approaches for this, (i) the shooting and bouncing (SBR) method and (ii) the image method [18]. In the former, the rays or ray tubes are emitted from the transmitter in all directions uniformly. Single ray propagation with time is traced, trying to establish whether the ray hits any facet or the receiver. The receiver is represented as a sphere with its radius depending on the angle between the shot rays and the length of the ray. If the closest ray collision is with a facet, the estimated reflected ray is traced on. If the ray hits the receiver, the ray energy is added to the received signal, and the ray is traced forward to find possible further interactions with the propagation environment. The computational complexity of the ray tracing method depends on the number of the transmitted rays and the number of facets describing the propagation environment, i.e. the scene. The second method, the image method [18], is based on the image theory. According to it, a plane conductive surface generates a new (mirrored) electromagnetic source, i.e. a virtual transmitter, which is applied to model a ray reflected once from the scene. Furthermore, each virtual source can generate another virtual source used for modeling the ray reflected twice from the scene, and so on. A ray reflected from a particular facet is represented as a direct ray from the virtual transmitter to the receiver. The algorithm complexity depends on the number of generated virtual sources, which is closely related to the number of facets and their positions and orientations in the scene.

The decision to apply the SBR method for ray tracing in rural areas originated from the following facts: (i) polarization of radio rays and ray diffuse scattering may be implemented, (ii) the SBR method is in close relation with the ray tracing algorithm in computer graphics and there exists quite a lot of efficient algorithms for ray tracing, (iii) the SBR algorithm may be efficiently parallelized and executed in parallel on the graphical processing units of contemporary computers, (iv) the number of transmitted rays may be reduced in the rural environment when the transmitter position is high above the environment, (v) the ray diffraction model can be efficiently modeled by adjusting the diameter of the received sphere according to the transmitter carrier frequency and finally (vi) the same method can be efficiently applied in urban areas and indoors.

The SBR method consists of three main tasks, namely (i) ray launching, (ii) tracing of the ray through the scene and (iii) deciding whether the ray hits the receiver or not. The basic concept can be found in [19]. Hereafter, we give a short overview of the SBR method and describe our modifications to speed up the whole process.

A. Ray Launching

Ray launching is a process where rays emanate from the transmitter location uniformly around the sphere. In order to achieve this, rays must be distributed uniformly around the sphere. Various approaches have been proposed in literature how to launch radio rays, but only regular polyhedrons fulfills the requirement of the large-scale and small scale uniformity [19]. The main property of regular polyhedrons is that the angles between neighboring rays launched from the polyhedron center to the directions of polyhedron vertices are equal around the sphere. This guarantees an uniform distribution of the emitted radio energy. Unfortunately, no regular polyhedron has more vertices than 20, which is not sufficient for the required precision in most cases. The problem can be partially solved by tessellating the facets of a regular polyhedron and extending the intersection points to the sphere surface. The regular polyhedron with 12 vertices, icosahedron, can be applied as a starting point. Each triangle of icosahedron is subdivided into 4 equal triangles, and after extending the intersection points onto the sphere border this results in a geodesic sphere with 42 vertices. The number of vertices in the next steps $N_{(i+1)}$ can be calculated by applying the following equation, starting with $N_0 = 12$:

$$N_{(i+1)} = 4(N_{(i)} - 2) + 2 \qquad i = 0,1,2,3,4,\ldots . \tag{1}$$

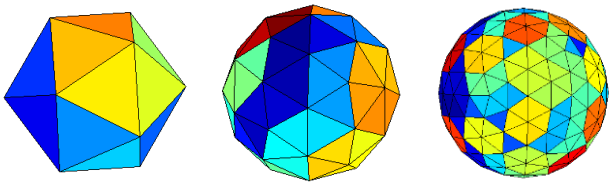

Fig. 8: An icosahedron and geodesic sphere for i = 0,1 and 2

An icosahedron and tessellated icosahedrons resulting in geodesic spheres for 42 and 162 vertices are depicted in Fig. 8. Two aberrations of the geodesic sphere occur in the described procedure of its generation, namely, (i) the rays emitted from the initial vertices of icosahedron are surrounded by five neighboring rays, while all the others with six and (ii) there is

an aberration in the angle separation for the rays neighboring on the initial rays from icosahedron vertices. These aberrations vanish at high tessellation frequencies. This is one reason for using the geodesic spheres with high number of vertices, e.g. 2562, 10242, 40962, 163842, etc. The second one is the resolution of rays at the borders of the analyzed area. When the area is large, the resolution of rays far from the transmitter is very poor. For example, for a geodesic sphere with 10242 vertices the ray resolution at a distance of 1 km is 34 m, which is far below the required resolution, while for a geodesic sphere with 163842 vertices it is 8.65 m.

B. *Tracing the Ray Trough the Scene*

When a ray is launched, the ray tracer should estimate whether the ray collides with the scene or receiver, or it misses them both. According to results presented in section 3, the relief can be represented by a huge number of triangles, and a thorough search through the complete list of triangles is time consuming. Speedup can be achieved by arranging triangles into a tree data structure. It has been proven that the Kd-Tree data structure is optimal for ray tracing [20,21] in computer graphics. Kd-Tree is an axis-aligned binary space partitioning (BSP) tree. The space is partitioned into two halves. Each half contains an equal number of elements. The two halves are processed recursively. The recursion stops when no half-space contains more than a preset number of elements, or when a predefined tree depth is achieved. At first glance, Kd-Tree looks similar to octree, which splits space in eight equal blocks and is for this reason inappropriate for ray tracing in rural areas. In contrast, the position of the splitting plane in Kd-Tree is not fixed and depends on the number of elements in each subspace.

An example of partitioning of the observation plane into two dimensional space using the Kd-Tree concept is illustrated in Fig. 9. The *split 1* on *axis x* divides the initial area into two unequal areal parts but with equal number of elements. Next, the algorithm proceeds on the left subspace, which is divided by *split 2* on *axis y* into two subspaces. The recursion on the left subspace is stopped because the stop condition is fulfilled. The algorithm returns to the right subspace and partitions it into two parts using *split 3* on *axis y*. After that, the algorithm continues on the upper part, finding *split 4*. When the stop conditions are fulfilled, it moves to the part below *split 3* and progresses its work calculating the position of *split 5*.

Extension of the algorithm to an arbitrary number of dimensions is straightforward. The algorithm is very robust if the observing area is unbalanced in one or more dimensions, which often appears when arranging triangles representing a flat rural area into a Kd-Tree.

After building the Kd-Tree, the scene ray intersections are estimated by traversing it. The flow chart of the algorithm is depicted in Fig. 10. At first it is checked if a ray intersects scene boundaries. If not, the algorithm stops and returns noObject. Next, by selecting appropriate child nodes, the subspace where the ray starts is determined first by traveling from the tree root to a tree leaf. The procedure is illustrated by the left loop in Fig. 10. The tree leaf contains a set of triangles, which are checked whether they intersect with the ray. If a ray triangle intersection is found, the traversing process stops, if not, the next subspace is checked based on the ray direction by popping the remaining tree nodes from the stack.

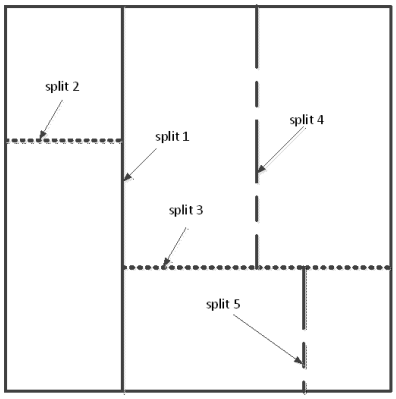

Fig. 9: Kd-Tree partitioning

Fig. 10: Traversing through Kd-Tree

The computational efficiency of the algorithm is in the selection of the child node which is processed and the child node which is pushed on the stack. The interested reader can

find the detailed description of the algorithm in [20,21]. In addition, an efficient method for finding the ray triangle intersections significantly improves the algorithm performance.

C. Ray Receiver Collision

Once a ray is launched, its contribution to the received signal has to be estimated. The common approach for this is using a reception sphere, which assumes uniform ray launching [19]. The receiver is represented by a sphere. If the ray intersects the sphere, it contributes to the received signal strength. The sphere radius depends on the ray characteristics, i.e. the angle between the neighbouring rays, and the length of the ray path. The problem is illustrated in Fig. 11. It is obvious that the reflected ray has a longer path and hence requires a reception sphere with a larger radius, compared to the sphere for the direct ray.

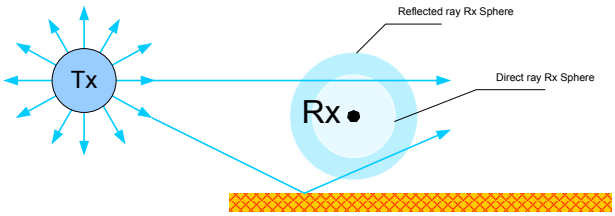

Fig. 11: Ray receiver collision

In three dimensional ray-tracing, a problem of double ray counting appears, which can be solved by (i) a weighting function proposed in [19] or (ii) by a software approach preventing the double counting of the same ray.

According to the method proposed in [19] the total electric fields is calculated by

$$\vec{E} = \sum_{i=1}^{N} f(x_i) V(\vec{x_R}\vec{x_i}) e^{j\phi_i} \vec{E_i}, \qquad (2)$$

where $\vec{x_R}$ is the receiver position, $\vec{x_i}$ the location of the i-th ray source, $\vec{E_i}$ vector of the electric field associated with the ray from the i-th source, ϕ_i phase of the electric field, N the number of ray sources in the scene, $f(x_i)$ weight function and $V(\vec{x_R}\vec{x_i})$ visibility function between the ray source and the receiver. The impact of the rays close to the edges of the reception sphere on the received power is lowered by the weighting function.

V. Results

The ray tracing algorithm was tested in an area of a typical rural macrocell, with a calculation resolution of 100 meters. The relief is represented by 38442 triangles when the *r.triangles* algorithm is applied, with only 5863 triangles if the relief data is pre-processed by the *r.refine* algorithm, and with 26348 triangles for the *r.Urban* algorithm. It is a predominantly flat area with some hills on the border of the cell (left and bottom). The transmitter is placed approximately in the middle of the area 20 meters above the relief.

The output of the ray tracer consists of the signal path loss in dB, mean delay in microseconds and RMS delay spread in microseconds. The results for 2562 rays launched from the transmitter are illustrated in Fig. 12, Fig. 13 and Fig. 14. The white spots denote areas with no signal. Those areas appear due to shadowing in the hilly areas. Because of the predominantly flat area, the delay spread is small and does not vary significantly.

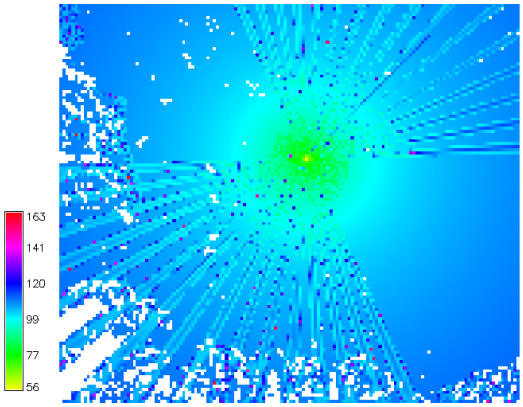

Fig. 12: Path loss in dB

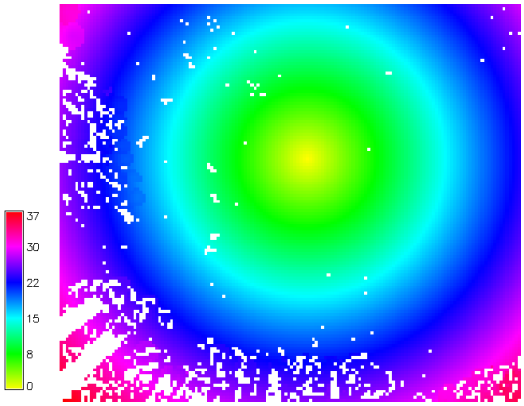

Fig. 13: Mean delay in μs

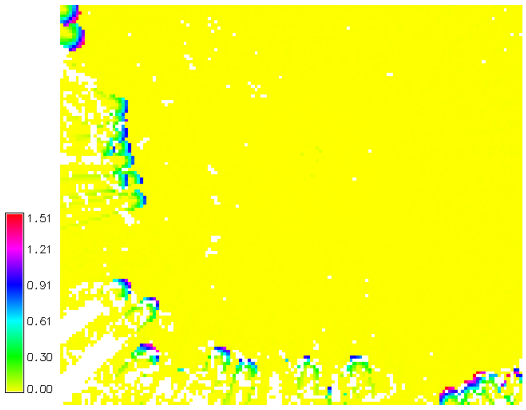

Fig. 14: RMS delay in μs

Table I contains the times required for the data and ray pre-processing for different numbers of initial rays, and the complete CPU times for the radio coverage calculation. The data and ray pre-processing is actually the time for calculating all the reflections of the initially emitted rays using the output data from the *r.refine, r.triangle* and *r.Urban* algorithms, and for storing the calculated rays into a list of rays. The data and ray pre-processing time is proportional to the number of transmitted rays. An increase of time by a factor of approximately four is observed comparing neighbouring rows. Comparing various tessellation algorithms (*r.triangles, r.refine, r.Urban),* i.e. comparing columns in Table I, shows that the algorithm speedup is proportional to the number of triangles generated by the tessellation algorithm. This conclusion is less obvious at low numbers of initially emitted rays due to some initialization procedures in the program.

The complete CPU time for the tested area and the resolution of 100 meters is also given in Table I. The CPU time to obtain results depends strongly on the number of receiver positions. Applying the *r.refine* triangulation speeds up the calculation by 600% for low numbers of rays launched and by 250% for higher numbers of launched rays. The processing time for low numbers of rays does not depend on the number of initially emitted rays. The reason is that the majority of the initially emitted rays do not intersect the scene, and the length of the list of rays after the data and ray pre-processing is nearly the same for 162, 642 and 2562 initially emitted rays.

The simulations are repeated for the urban area, Table II, where the number of triangles describing the city centre is much higher. Buildings are high compared to the area of the radio coverage calculation. Consequently, the CPU times in Table II and the relations between its columns and rows are changed, but the main conclusions from the paragraphs above are still valid.

Table I. Ray launching and CPU time (in seconds) for flat rural area

number of rays	*r.triangle*	*r.refine*	*r.Urban*
	Data & Ray preprocessing		
162	2.5	0.7	2.2
642	8.8	1.9	6.1
2562	33.1	5.7	23.2
10242	129.5	20.6	90.0
	Complete CPU time		
162	356.6	55.4	242.2
642	337.1	60.1	231.8
2562	363.3	86.7	259.1
10242	555.8	202.9	423.6

Table II. Ray launching and CPU time (in seconds) for urban area

number of rays	*r.triangle*	*r.refine*	*r.Urban*
	Data & Ray preprocessing		
162	30.7	7.3	6.8
642	111.3	26.3	24.2
2562	435.8	100.8	91.0
10242	1720.9	405.4	364.6
	Complete CPU time		
162	88362.6	18864.4	17062.9
642	80307.8	20654.0	17288.2
2562	74432.6	18633.7	17354.3
10242	80839.2	20121.9	18998.7

VI. Conclusion

In this paper, a description of the shooting and bouncing ray approach for 4G Radio network planning has been described. The efficiency of the method has been analyzed in terms of the required CPU time for the data and ray preprocessing and the complete CPU time for calculation of the area coverage with radio signal. The method can be further improved by adding the data about the material type and its roughness for each triangle. This would be the base for the introduction of ray scattering into the model, as well as for the possibility of calculation of radio wave penetration into the buildings. An efficient method for radio ray diffraction on horizontal and vertical obstructions should also be included in the model to correctly estimate the strength of the radio signal.

The ray tracing method may be significantly speeded up by using graphical processing units, which are optimized and widely applied for ray tracing in computer graphics. According to some preliminary tests, parallel GPU processing may speed up the method up to 1000 times.

References

[1] M. Hata, "Empirical Formula for Propagation Loss in Land Mobile Radio Services," *IEEE Transaction on Vehicular Technology, v*ol. 29, no. 3, August 1980.

[2] J.-E. Berg, "A Recursive Method For Street Microcell Path Loss Calculations," *in Proc. Sixth IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC'95.*, 27-29 Sep 1995, vol. 1, pp. 140-143.

[3] J.-E. Berg, H. Holmquist, "An FFT multiple half-screen diffraction model," *in Proc. IEEE VTC'94*, Stockholm, Sweden, pp. 195-199, 1994.

[4] S. Jimaa, C. Kok Keong, Y. Chen, Y. Alfadhl, "LTE-A an Overview and Future Research Areas," *in Proc. Second International Workshop on the Performance Enhancements in MIMO-OFDM Systems*, Shanghai, China, 2011.

[5] Y. Corre, and Y. Lostanlen, "Three-Dimensional Urban EM Wave Propagation Model for Radio Network Planning and Optimization Over Large Areas," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 7, 2009.

[6] G. E. Athanasiadou, I. J.Wassell, "Comparisons of Ray Tracing Predictions and Field Trial Results for Broadband Fixed Wireless Access Scenarios," *WSEAS Transactions on Communications*, issue 8, vol. 4, August 2005, pp. 717-721.

[7] Directory of wireless system planning tools, http://rsl.ece.ubc.ca/planning.html.
[8] A. Hrovat, T. Javornik, S. Plevel, R. Novak, T. Celcer, I. Ozimek, "Empirical path Loss Model and WiMAX Field Measurements in Urban and Suburban Environment," *WSEAS Transactions on Communications*, issue 8, vol. 5, August 2006, pp. 1328-1334.
[9] Q-Rap home page, http://www.qrap.org.za/home.
[10] A. Hrovat, I. Ozimek, A. Vilhar, T. Celcer, I. Saje, and T. Javornik, "Radio Coverage Calculations of Terrestrial Wireless Networks using an Open-Source GRASS System," *WSEAS Transactions on Communications*, vol. 9, no. 10, 2010.
[11] GRASS-RaPlaT home page, http://www-e6.ijs.si/en/software/grass-raplat.
[12] GRASS Development Team, 2010. Geographic Resources Analysis Support System (GRASS) Software, Open Source Geospatial Foundation. http://grass.osgeo.org.
[13] A. Vilhar, A. Hrovat, I. Ozimek and T. Javornik, "Efficient Open-source Ray-Tracing methods for rural environment," *in Proc. 16th WSEAS International Conference on Computers*, Kos Island, Greece, July 15-17, 2012.
[14] J. Todd, and L. Toma, "Scalable Raster-to-TIN simplification," *in Proc. Free and Open Source Software for Geoinformatics (FOSS4G 2006)*, 2006.
[15] M. Heller, "Triangulation algorithms for adaptive terrain modeling," *in Proc. 4th Intl. Symp. On Spatial Data Handling*, Zürich, 1990.
[16] B. Delaunay, "Sur la sphère vide," *Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk*, 1934.
[17] J. R. Shewchuk, "Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator," *in Applied Computational Geometry: Towards Geometric Engineering*, vol. 1148 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, May 1996, pp. 203-222.
[18] M. F. Cátedra, and J. Pérez-Arriaga, *Cell Planning for Wireless Communications*, Artech House Boston, London, 1999.
[19] G. Durgin, N. Patwari, T. S. Rappaport, "An advanced 3D ray launching method for wireless propagation prediction," *in Proc. IEEE 47th Vehicular Technology Conference*, Phoenix, USA, 1997.
[20] V. Havran, "Heuristic Ray Shooting Algorithms," Ph.D. Thesis, Czech Technical University, Faculty of Electrical Engineering, Department of Computer Science and Engineering.
[21] I. Wald, and V. Havran, "On building fast kd-trees for ray tracing, and on doing that in O(Nlog N)," *in Proc. 2006 IEEE Symposium on Interactive Ray Tracing*, 2006.

Andrej Vilhar was born in Ljubljana, Slovenia, in 1979. He received the B.Sc. and the Ph.D. degree in Electrical Engineering from the University of Ljubljana, Slovenia, in 2004 and 2009, respectively.

He spent one year at the French aerospace lab ONERA, Toulouse, as a post-doc researcher. He is currently a researcher at the Department of Communication Systems of the Jožef Stefan Institute. His previous research interests include network modeling and mobility management in the IPv6 networks. Currently, he is focusing on radio signal propagation and channel modeling, with the emphasis on terrestrial network planning and on the microwave satellite signal measurements.

Andrej Hrovat was born in Novo mesto, Slovenia, in 1979. He received a B.Sc. and M.SC. in Electrical Engineering from the University of Ljubljana, Slovenia, in 2004 and 2008, respectively. He obtained a Ph.D. degree in Electrical Engineering from the Jožef Stefan International Postgraduate School, Slovenia, in 2011.

He is currently a researcher in the Department of Communication Systems of the Jožef Stefan Institute and assistant at the Jožef Stefan International Postgraduate School. His research interests include radio signal propagation, channel modeling, terrestrial and satellite fixed and mobile wireless communications, radio signal measurements and emergency communications.

Igor Ozimek was born in Ljubljana, Slovenia, in 1957. He received the B.Sc., the M.Sc., and the Ph.D. degree in Electrical Engineering from the University of Ljubljana, Slovenia, in 1980, 1988 and 1993, respectively.

He spent six months at the University of Westminster, London, UK, as a visiting scientist. He is currently a researcher in the Department of Communication Systems at the Jožef Stefan Institute. His research interests include digital communications, GPU processing, and computer networks.

Tomaž Javornik was born in Kočevje, Slovenia, in 1962. He received the B.Sc., the M.Sc., and the Ph.D. degree in Electrical Engineering from the University of Ljubljana, Slovenia, in 1987, 1990 and 1993, respectively.

He spent six months at the University of Westminster, London, UK, as a visiting scientist. He is currently a senior researcher in the Department of Communication Systems at the Jožef Stefan Institute and assistant professor in the Jožef Stefan International Postgraduate School. His research interests include radio signal propagation channel modeling in terrestrial and satellite communication systems, adaptive coding and modulation, adaptive antenna and MIMO systems, wireless optical communication, cooperative communication, adaptive coding and modulation, relay communication and self-organizing networks.