A NEW STRATEGY FOR INDOOR PROPAGATION FAST COMPUTATION WITH MR-FDPF ALGORITHM.

G. de la Roche, R. Rebeyrotte
Sygmum
Lyon, France
email: gdelaroche@sygmum.com

K. Runser
ARES, INRIA
Villeurbanne, France
email: katia.runser@insa-lyon.fr

J-M. Gorce
CITI, INSA Lyon
Villeurbanne, France
email: jean-marie.gorce@insa-lyon.fr

ABSTRACT
The multi-resolution frequency domain ParFlow (MR-FDPF) algorithm has been developed since 2001 for the purpose of Indoor wave propagation simulation. This algorithm derives from a TLM like formalism, and implements a multi-resolution (MR) approach to compute efficiently full coverage of several access points (AP). This approach is well-designed for a wireless LAN planning because the computation load required to process an AP candidate is fast. The heart of the proposed approach holds in the MR-nodes that are obtained by gathering recursively pixels into bricks. This requires a MR binary tree structure which should be adapted to the environment. In this paper, different strategies are evaluated to compute this binary tree. Computation load and memory consumption are evaluated on a real example. It is shown that an appropriate choice of the binary tree may decrease significantly the computation load associated with the method.

KEY WORDS
Indoor propagation, Finite difference frequency domain, multi-resolution, TLM, ParFlow, wireless LAN.

1 Introduction

The generalization of Indoor radio systems (Bluetooth, WiFi , UMTS, ...) deployed in personal or professional environments is exploding. This calls for more efficient design and planning tools. Because Indoor propagation suffers numerous reflections and diffractions the prediction of wave propagation is difficult to assess. This problem has been tackled for a couple of years and several tools have been developed. The heart of these tools is the wave propagation engine which aims to predict the behavior of waves in realistic environments. Two kinds of wave propagation engines are widely used: empirical and deterministic approaches.

- Empirical approaches (see for instance [1]) are based on statistic models of propagation. These methods allow to predict the average behavior of waves in typical environments but surely not to provide accurate predictions in closed areas.
- Deterministic approaches have been developed to improve the propagation prediction in small dense urban areas and indoor environments [2, 3], for which empirical approaches failed. By analogy with the light propagation, these approaches are based on geometric optic (G.O.) laws and computation engines take their origin in the world of image rendering. In these methods, rays are launched, attenuated and reflected depending on the obstacles they meet during their travel. Because these approaches are based on a deterministic model, they provide more accurate results than empirical approaches. They suffer however of a high computation load for wide open areas.

Compared to these approaches, only few works tackled the propagation simulation with finite difference approaches. The reason is of course the computation load which is particularly high in these approaches. However in 1998 Luthi et al.[4, 5] have proposed a new discrete approach, ParFlow (Partial Flows), based on the Cellular Automata formalism. Although the theoretical background is different, ParFlow when applied to electromagnetic waves, is similar to the Transmission Line Matrix method [6] with specific dispersion nodes. In order to increase the computational efficiency, Luthi [4] has proposed an implementation on parallel architectures, taking advantage of the Cellular Automata structure. It was applied for GSM mobile systems in urban environments. The main advantage of this approach is that all propagation effects including reflection and diffraction are naturally taken into account. But the required resolution is in turns very high and some approximations are required to perform the propagation simulation under a reasonable time.

At our knowledge, this method has never been applied such as, in Indoor environments. In 2001 [7], we have derived in the frequency domain an efficient algorithm to solve this problem in two dimensions (2D); see section 2. This algorithm is based on a multi-resolution approach. Conventional TLM nodes are firstly gathered two by two leading to nodes of higher level. These nodes can be gathered once more leading to other nodes, and so on until only one node encompasses all the environment. A binary tree is thus built, gathering progressively all the pixels of the image. To build this tree, ascending or descending approaches can be used, as detailed in section 3. In ascending approaches, nodes are gathered two by two recursively, while in descending ones, each node is settled into two children, starting from the head node. The way the binary tree is

built plays a fundamental role on the computation load and memory consumption of the method, as detailed in section 4. This paper focuses on this aspect and proposes an efficient heuristic to optimize computation and memory loads.

2 MR-FDPF Formalism

2.1 The Conventional Node in Frequency Domain

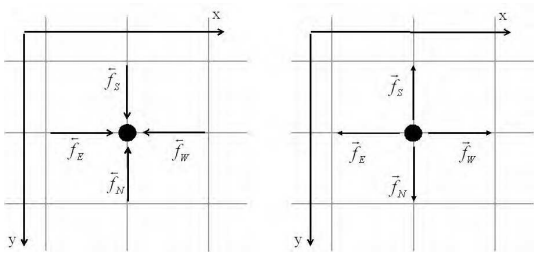

Figure 1. 4 inward flows (a) and 4 outward flows (b) are associated with each pixel.

In a standard frequency domain approach starting either from TLM or ParFlow theory, the pixels are nodes over which the field is computed, and each node is linked by branches with its neighbors. The TLM or ParFLow equations then reflect the way flows propagate on this grid. Four inward and four outward flows are associated with each node as illustrated in Figure 1.

The propagation of a source is managed by the transition line matrix providing the outward flows of a node $m = \{i, j\}$ according to the inward flows:

$$\overrightarrow{F}(m) = \Sigma(m) \cdot \overleftarrow{F}(m) + S(m) \tag{1}$$

where $\Sigma(m)$ is the local scattering matrix and the flow vectors are given by

$$\overleftarrow{F}(m) = \begin{pmatrix} \overleftarrow{f_E}(i,j) \\ \overleftarrow{f_W}(i,j) \\ \overleftarrow{f_S}(i,j) \\ \overleftarrow{f_N}(i,j) \end{pmatrix}; \quad \overrightarrow{F}(m) = \begin{pmatrix} \overrightarrow{f_E}(i,j) \\ \overrightarrow{f_W}(i,j) \\ \overrightarrow{f_S}(i,j) \\ \overrightarrow{f_N}(i,j) \end{pmatrix} \tag{2}$$

The links between adjacent node flows are given by

$$\begin{aligned} \overrightarrow{f_E}(i,j) &= \overleftarrow{f_E}(i+1) \\ \overrightarrow{f_W}(i,j) &= \overleftarrow{f_W}(i-1) \\ \overrightarrow{f_S}(i,j) &= \overleftarrow{f_S}(i,j) \\ \overrightarrow{f_N}(i,j) &= \overleftarrow{f_N}(i,j) \end{aligned} \tag{3}$$

The FDPF algorithm can be implemented in a modified cellular automata framework [8] leading to

Algorithm 2.1

Initialization

$\forall m$, *Set* $F_{acc}(m) = \overrightarrow{F}(m) = S(m)$

Do *(until stability is reached)*

Update inward flows: $\forall m$, $\overleftarrow{F}(m) = \mathcal{N}(\overrightarrow{F}(m))$

Accumulate : $\forall m$, $F_{acc}(m) = F_{acc}(m) + \overleftarrow{F}(m)$

Compute outward flows : $\forall m$, $\overrightarrow{F}(m) = \Sigma(m) \cdot \overleftarrow{F}(m)$

Until convergence ($\overleftarrow{F}(m) \ll F_{acc}(m)$)

with

$$\mathcal{N}(\overrightarrow{F}(m)) = \begin{pmatrix} \overrightarrow{f_E}(i-1,j) \\ \overrightarrow{f_W}(i+1,j) \\ \overrightarrow{f_S}(i,j-1) \\ \overrightarrow{f_N}(i,j+1) \end{pmatrix} \tag{4}$$

2.2 The Multi-Resolution Node: MR-Node

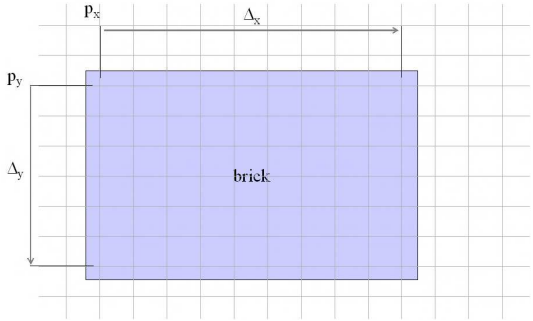

Figure 2. A MR-node is defined as a rectangular set of pixels.

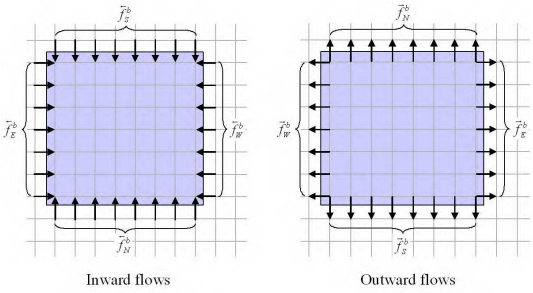

Figure 3. Inward flows (a) and Outward flows (b)are associated with each MR-node.

The multi-resolution FDPF is developed on the bloc concept derived for the first time in 2001 [7], in which a multi-resolution node (MR-node) has been defined as a rectangular set of aggregated pixels. Shlepnev [9] proposed

a similar concept associated with the TLM formalism and called the brick. Indeed, the MR-node is nothing else than a rectangular piece of space. The MR-node is a rectangular set of pixels and is defined by its size ($\overrightarrow{\Delta} = (\Delta_c, \Delta_r)$), and the position of its top-left pixel ($\overrightarrow{p} = (p_x, p_y)$) : $B(\overrightarrow{\Delta}, \overrightarrow{P})$ (see Fig. 2). All the flows belonging to a MR-node b are gathered into three vectors. In one hand, inward and outward flow vectors gather the flows located on the bounds of the MR-node as illustrated in Fig.8 a and b. They are given by:

$$\overleftarrow{F}(b) = \begin{pmatrix} \overleftarrow{f_E}(b) \\ \overleftarrow{f_W}(b) \\ \overleftarrow{f_S}(b) \\ \overleftarrow{f_N}(b) \end{pmatrix} \quad ; \quad \overrightarrow{F}(b) = \begin{pmatrix} \overrightarrow{f_E}(b) \\ \overrightarrow{f_W}(b) \\ \overrightarrow{f_S}(b) \\ \overrightarrow{f_N}(b) \end{pmatrix} \tag{5}$$

In the other hand, the inner flow vector $\overset{o}{F}(b)$ gather all the other flows included in the MR-node: these flows are located on branches connecting two pixels of the brick and is given by

$$\begin{aligned} \overset{o}{F}(b) = [&\overrightarrow{f_E}(0,0), \overrightarrow{f_E}(0,1), \ldots, \overrightarrow{f_E}(N_r - 1, Nc - 2), \\ &\overrightarrow{f_W}(0,1), \overrightarrow{f_W}(0,2), \ldots, \overrightarrow{f_W}(N_r - 1, Nc - 1), \\ &\overrightarrow{f_S}(0,0), \overrightarrow{f_S}(0,1), \ldots, \overrightarrow{f_S}(N_r - 2, Nc - 1), \\ &\overrightarrow{f_N}(1,0), \overrightarrow{f_N}(1,1), \ldots, \overrightarrow{f_N}(N_r - 1, Nc - 1)]^t \end{aligned} \tag{6}$$

where the indices refer to the position inside the MR-node.

The scattering matrix of a MR-node B provides the links between outward and inward flows, as for the usual node. This scattering matrix may be expended into 16 submatrices according to the direction of concerned inward and outward flows, leading to

$$\Sigma(B) = \begin{bmatrix} \sigma_{EE}(B) & \sigma_{EW}(B) & \sigma_{ES}(B) & \sigma_{EN}(B) \\ \sigma_{WE}(B) & \sigma_{WW}(B) & \sigma_{WS}(B) & \sigma_{WN}(B) \\ \sigma_{SE}(B) & \sigma_{SW}(B) & \sigma_{SS}(B) & \sigma_{SN}(B) \\ \sigma_{NE}(B) & \sigma_{NW}(B) & \sigma_{NS}(B) & \sigma_{NN}(B) \end{bmatrix} \tag{7}$$

2.3 The Tree Structure

Let now a MR-node C, be settled into two child MR-nodes A and B, according to Fig.4. The inner flows associated with this bloc are now not all the flows inside the node but only the exchange flows between nodes A and B as illustrated in Fig.4.

It may be shown that the scattering matrix of MR-node C can be computed from those of its children [8], according to

$$\Sigma(C) = U(C) \cdot I(C) \cdot D(C) \tag{8}$$

where $U(C)$, $I(C)$ and $D(C)$ are respectively the upward, the downward, and the inner matrices associated with MR-node C and given by

$$D(C) = \begin{pmatrix} [0] & \sigma_{WW}(B) & [0] & \sigma_{WS}(B) & [0] & \sigma_{WN}(B) \\ \sigma_{EE}(A) & [0] & \sigma_{ES}(A) & [0] & \sigma_{EN}(A) & [0] \end{pmatrix} \tag{9}$$

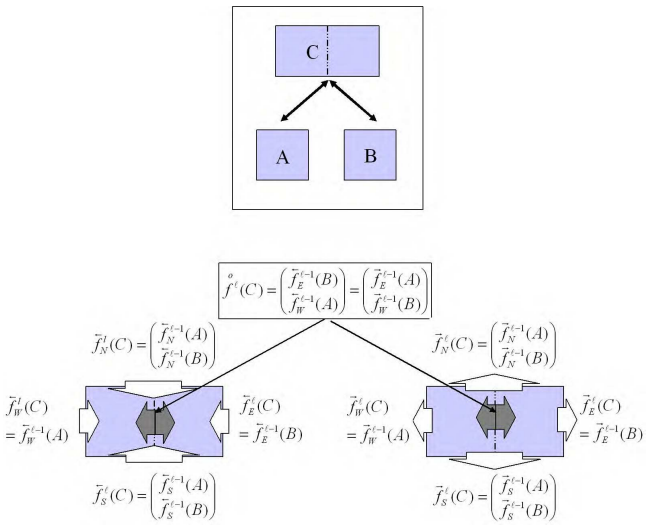

Figure 4. the flows of a MR-node C are related to the flows of its two child bricks.

and

$$I(C) = \begin{pmatrix} t_{WW}(C) & t_{WE}(C) \\ t_{EW}(C) & t_{EE}(C) \end{pmatrix} \tag{10}$$

with
$t_{EE}(C) = (I_d - \sigma_{EW}(A) \cdot \sigma_{WE}(B))^{-1}$,
$t_{WW}(C) = (I_d - \sigma_{WE}(B) \cdot \sigma_{EW}(A))^{-1}$,
$t_{WE}(C) = \sigma_{WE}(B) \cdot t_{EE}(C)$,
$t_{EW}(C) = \sigma_{EW}(A) \cdot t_{WW}(C)$.
and

$$U(C) = \begin{pmatrix} [0] & \sigma_{EE}(B) \\ \sigma_{WW}(A) & [0] \\ [0] & \sigma_{SE}(B) \\ \sigma_{SW}(A) & [0] \\ [0] & \sigma_{NE}(B) \\ \sigma_{NW}(A) & [0] \end{pmatrix} \tag{11}$$

It should be noted that these matrices correspond to a vertical cutting of MR-node C. For an horizontal cutting,these matrices are similar but with permutation of indices E (East) and S (South) in one hand, and of indices W (West) and N (North) in the other hand.
It can be emphasized that only $I(C)$ and $\Sigma(C)$ should be computed and stored for each MR-node, the others pointing directly to parts of its child MR-nodes scattering matrices.

A binary tree thus can be built, as described in section 3. A regular structure which is a specific case, leads to a pyramidal tree as illustrated in Fig.5. In such a structure, a (ℓ)-level MR-node contains two $(\ell - 1)$-level MR-nodes.
The pre-processing phase consists in building this tree structure, and computing the scattering matrices associated with each MR-node.

2.4 Computing a Source Coverage

Once the binary tree is created and scattering matrices computed, the propagation can be done in two steps : the upward and the downward. The upward step starts from the

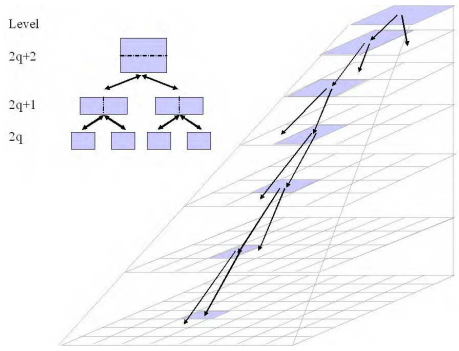

Figure 5. The regular multi-resolution approach is based on a regular binary tree for assembling MR-nodes.

source node, and computes the propagation inside its father MR-node. When its outward flows are computed, the father node stands for the new source node. The propagation is then computed in its own father node, and so on until the head node of the binary tree is reached. The two following elementary operations are done in each father of source node

- the steady-state inner flows of father MR-node (C) propagate inside C, as in an isolated world, as illustrated in Fig.7(a). Steady-state flows are then given by

$$\overset{o}{F}(C) = I(C) \cdot \begin{pmatrix} \overrightarrow{f_W}(B) \\ \overrightarrow{f_E}(A) \end{pmatrix} \quad (12)$$

- The second propagates these steady-state inner flows toward outward flows as illustrated in Fig.6(b).

$$\overrightarrow{F}(C) = U(C) \cdot \overset{o}{F}(C) \quad (13)$$

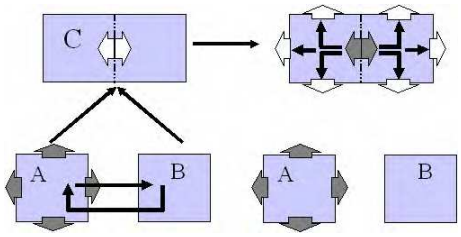

Figure 6. Upward step. In the left, the MR-node A is a source and Est flow propagates inside C; the steady-state is computed. In the right, the inner flows are propagated to outward flows providing the flows of the MR-node C.

The downward step implements the propagation of inward flows to inner flows independently in each MR-bloc, according to Fig.7(c) and to

$$\begin{pmatrix} \overleftarrow{f_W}(A) \\ \overleftarrow{f_E}(B) \end{pmatrix} = D(C) \cdot \overleftarrow{F}(C) + \begin{pmatrix} \overrightarrow{f_W}(B) \\ \overrightarrow{f_E}(A) \end{pmatrix} \quad (14)$$

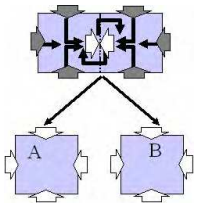

Figure 7. Downward step. Inward flows of MR-node C propagate inside C leading to inward flows of MR-nodes A and B.

3 Algorithms for Building Binary Trees

3.1 Constraints

A regular binary tree is the easiest to be built. However, it is shown in this paper that it is neither the only way, neither the most efficient as described in this section.
The efficiency is considered with respect to four criteria:

- The computation time of the pre-processing step, dedicated to the computation of the two matrices $\Sigma()$ and $I()$ associated with each MR-node.
- The memory needs, to store both matrices.
- The computation time needed to compute the coverage of a source over the whole environment at the pixel resolution.
- The computation time needed to compute the coverage of a source but at a rough resolution.

It is indeed possible to reduce slightly more the computation time associated with the source propagation by stopping the downward propagation at MR-nodes corresponding to free-space homogeneous nodes. A free-space homogeneous MR-node is defined as a node containing only air cells. For these nodes, the mean received power of all included cells is evaluated directly from the inward flows. This reduces the amount of downward propagations performed over the whole space and the resolution used for the coverage prediction is adapted to the heterogeneity of the environment. In wide free space areas, the mean received power is computed over wide MR-nodes, while in dense areas (numerous obstacles) the received power is computed over smaller MR-nodes. This is illustrated in section 4.

It appears clearly that the main computation load and memory consumption in the pre-processing phase is due to the computation of scattering matrices. Thus, a computation and memory loads are theoretically associated with each MR-node. And higher is the MR-node size, higher are the loads. It is obvious to show that the computation load evolves in $O(N^3)$ while the memory consumption evolves in $O(N^2)$, where N is the length of exchange flow vectors between child nodes. However this can be reduced when considering that some MR-nodes are sometimes similar in

the environment. Two MR-nodes are said similar if they have same size and contain exactly the same cells. In this case, their matrices are exactly the same and they can be computed and stored only once. It is therefore interesting to build a binary tree privileging the reuse of same MR-nodes. For this purpose, a brick is defined as a model of MR-node. A MR-node is thus a node having a father, two children, flow vectors and a reference to its model : the brick. The brick is a MR-node model having two children, not an explicit father but several unknown fathers, no flows but scattering matrices.

The aim of the preprocessing phase is twofold: building the binary tree (very fast) and building the brick database with the computation of associated scattering matrices.

3.2 Descending Approach

An ascending approach would consist to start from the pixels, and to gather them recursively until all are gathered into the head node of the binary tree. The advantage would be to take into account more accurately the local arrangement of walls and others obstacles. However, an ascending approach cannot warrant both to preserve the rectangular form of MR-nodes and the binary nature of the tree. This would lead globally to an overhead computation and memory loads.

In the opposite, the descending approach tackles the problem from the head node, and divide it into two children MR-node. Then each child is itself divided, and so on until the unitary nodes are reached. Several empiric algorithms can be developed to decide how each MR-node is divided. Keeping in mind the aim of reducing the memory needs, a first idea could be to try to minimize the number of MR-nodes. It is in this case obvious to show that the best approach consists in cutting each MR-node along a line in the middle of its higher length. This leads to the regular binary-tree. However this is probably not the most efficient cutting. Indeed, memory and computation loads depends rather on the number of bricks than on the number of MR-nodes. It appears indeed more efficient to try to reduce not the number of MR-nodes but rather the number of bricks (reference MR-nodes).

In this case, another proposal is to try to align cuts in MR-nodes along walls and obstacles, to reach as soon as possible homogeneous MR-nodes. To implement this approach, the following rules are followed:

- Select the longer side of a MR-node, (N pixels).
- Compute the number of discontinuities $D(i)$ for each possible splitting line l_i; $\forall i \in [1; N-1]$. In the case of a vertical cut, the number of discontinuities $D(i)$ is given by the number of lines j for which the material indices $n(i,j)$ and $n(i+1,j)$.

$$D(i) = \sum_{j=0}^{M-1} \left(n(i-1,j) \neq n(i,j)\right) \qquad (15)$$

- Split the bloc at the index i_m such as $D(i)$ is maximum.

$$i_m = \underset{i\in[1;N-1]}{argmax} D(i) \qquad (16)$$

It however appeared that splitting MR-nodes along the highest discontinuity increases sometimes strongly the preprocess computation load. Indeed, this approach favors higher MR-nodes than the regular splitting. We propose a tradeoff between both approaches by weighting the previous criteria as follows:

$$i_m = \underset{i\in[1;N-1]}{argmax} D(i) \cdot C(i) \qquad (17)$$

where

$$C(i) = 1 - \left|\frac{i-co}{co}\right|^p \qquad (18)$$

with co the half-length of the node. p is given by

$$p = \begin{cases} 0 & \text{if } N < L, \\ K & \text{if } N \geq L \end{cases} \qquad (19)$$

where K and L are settings parameters. Their meaning is the following. For MR-nodes having sides smaller than L, the splitting is discontinuity based only. For larger MR-nodes the splitting line is chosen as a tradeoff between discontinuities and distance from the middle of the node.

3.3 Implementation

The method has been implemented on a JAVA virtual machine for portability purposes. Matrix computations are down using the COLT library developed by the CERN, Geneva. However, because of the high number of matrix inversion and multiplications done during the preprocessing phase, and because JAVA is not optimized for matrix computation, a Java interface (JNI) has been developed to perform calls to the BLAST (Basic Linear Algebra Subprogram)library. The BLAST library is a freely distributed library including procedures for intense matrix calculus which can be compiled specifically for each system. This approach has allowed us to save computation time. The whole computation time for the preprocessing phase decreased by a factor of 8 for a 500×500 pixels. Furthermore, higher is the environment size, higher is the saving time factor. The following results have been obtained on a PC pentium 2GHz.

4 Results

The test environment of about 120m x60m contains a building floor in which a wLAN has been deployed. The simulations are done with resolution step of 20cm. In order to respect the resolution criteria given in [4, 8] and requesting that the resolution should be at least six times lower than the wavelength, the simulation frequency used is not the

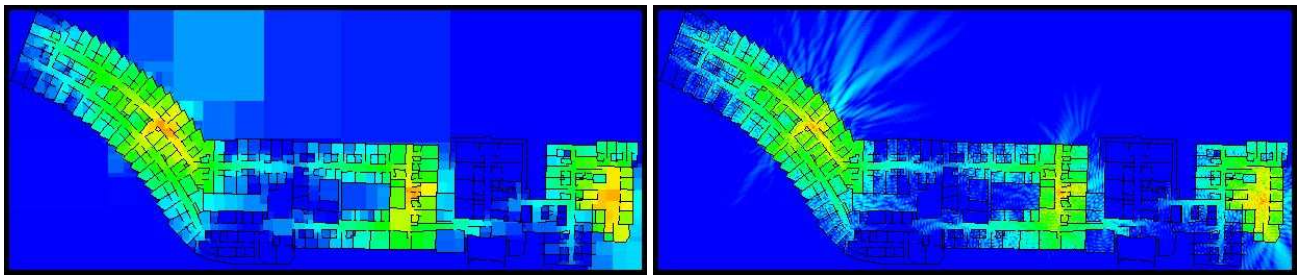

Figure 8. Coverage maps for 3 access points obtained at homogeneous bloc level (a) and at pixel level (b). Red, green and blue stand respectively from -30dBm down to -100dBm.

Table 1. computation and memory loads

	method		
	regular	optimal (L=32,K=6)	disc.
memory	48 MO	62 MO	87 MO
CPU time	17 s	11 s	32 s
brick H20	37%	55%	54%

true frequency (2.4GHz), but rather a lower one. The impact of this approximation has been detailed in [10]. Mean errors between measurements and simulations were in the order of 5dB.

The final results concerning the preprocessing phase are summarized in table A. The lower memory use is obtained with the regular tree. However, it corresponds to a small number of homogeneous bricks. Only 37% of the whole free-space is represented by large homogeneous nodes (size $>$ 20pixels). In the opposite, with the discontinuity base splitting algorithm the CPU time increases by a factor of 3 and the memory needs by a factor of 2. The modified approach makes a trade-off between both approaches. With the use of the modified approach, the propagation of a source lasts 450 ms to reach the homogeneous level and 8 seconds to reach the pixel level. the resulting coverage maps are provided in Fig.4.

5 Conclusion

In this paper, implementation and computational performances of the MR-FDPF approach are presented. An original approach is proposed to optimize the computation time required to compute the coverage of access points in Indoor environments. With the adaptive tree built as proposed the preprocess time is about few seconds and the propagation to the homogeneous bloc is under one second for the tested environment (600x300 pixels). This is very useful to implement planning algorithms for which hundreds AP positions have to be tested.

References

[1] K. Cheung, J. Sau, and R.D.Murch, "A new empirical model indoor propagation prediction," *IEEE Trans. Vehic. Tech.*, 1997.

[2] R. Valenzuela, "Ray tracing approach to predicting indoor wireless transmission." in *proc. of 43rd IEEE VTC*, May 1993, pp. 214–218.

[3] D. Lu and D. Rutledge, "Indoor wireless channel modeling from 2.4 to 24ghz using a combined e/h-plane 2d ray tracing method." in *Int. Symp. on Ant. and Prop.*, Monterey, CA., 2004.

[4] P. O. Luthi, "Lattice wave automata," Ph.D. dissertation, Universite de Geneve, Genéve, Switzerland, march 1998.

[5] B. Chopard, P. Luthi, and J. Wagen, "A lattice boltzmann method for wave propagation in urban microcells," in *IEEE Proceedings - Microwaves, Antennas and Propagation*, vol. 144, 1997, pp. 251–255.

[6] W. Hoefer, "The transmission line matrix method - theory and applications," *IEEE Trans. on Mic. Th. Tech.*, vol. 33, no. 4, pp. 882–893, 1985.

[7] J.-M. Gorce and S. Ubeda, "Propagation simulation with the parflow method : fast computation using a multi-resolution scheme." in *IEEE 54th VTC*, 2001.

[8] J.-M. Gorce, E. Jullo, and K. Runser, "An adaptative multi-resolution algorithm for 2D simulations of indoor propagation," in *Proc. 12th ICAP*. Exeter, UK: IEE, April 2003, pp. 216–219.

[9] Y. Shlepnev, "Trefftz-type brick finite elements for electromagnetics," in *proc. of the annual review of progress in applied computational electromagnetics*, Monterey, March 2001.

[10] K. Runser, G. de la Roche, and J.-M. Gorce, "Assessment of a new indoor propagation prediction model based on a multi-resolution algorithm," in *Proc. VTC Spring 2005*, May 2005.