

The IlmProp

Giovanni Del Galdo and Jörg Lotze

2005

Contents

1	About	3
1.1	What is the IlmProp?	4
1.1.1	What does it do?	4
1.1.2	What is the current development status?	4
1.1.3	How does it work?	5
1.1.4	What does it NOT do?	5
1.2	Typical Applications	6
1.2.1	Measurement-based applications	6
1.2.2	non Measurement-based applications	7
1.3	Requirements	8
1.4	General License Agreement and Lack of Warranty	8
1.5	Future Updates	9
1.6	Credits	9
2	Getting Started	11
2.1	Downloading and Installing IlmProp	12
2.1.1	Installing	12
2.2	Quick Walkthroughs	12
2.2.1	Generate a new geometry and visualize it	12
2.2.2	Generate the channel matrix, visualize it, and save it	13
2.2.3	Modify an existing geometry	13
2.2.4	Making a brand new geometry	13
2.3	Templates	14
3	Frequently Used Terms	15
4	Coordinate Systems	17
5	Data Structures	19
5.1	The struct Geometry	20
5.1.1	Geometry.Rx and Geometry.Tx	20
5.1.2	Geometry.loss_coefficients	23
5.1.3	Geometry.S	23

5.1.4	Geometry.Obstacles	23
5.1.5	Geometry.Obstacles	24
5.2	The struct data	24
5.2.1	data.H, data.H_DMC, data.H_LOS, and data.H_NLOS	24
5.3	The EADF	25
6	The Graphical User Interfaces	26
6.1	The main IlmProp GUI	26
6.2	The Geometry Selection	27
6.3	The Geometry Explorer	31
6.4	The Array Explorer	32
6.4.1	Array Explorer - geometry	32
6.4.2	Array Explorer - directivity	34
6.4.3	Array Explorer - all beams	35
6.4.4	Array Explorer - directivity in az	37
6.5	The Bello Explorer	38
7	The Channel Generation	40

Chapter 1

About

1.1 What is the IlmProp?

The IlmProp is a flexible time variant frequency selective channel propagation modelling tool for multi-user MIMO systems. Its original concept and development are by Giovanni Del Galdo, who remains curator of the IlmProp Project. Jörg Lotze joined later giving significant contributions to its development. See 1.6 for a comprehensive list of people who helped in the development of the IlmProp.

1.1.1 What does it do?

The main scope of the IlmProp is the generation of Channel Impulse Responses (CIR) as a sum of propagation rays. The channels are computed in delay time and frequency domain. The rays are defined by a three-dimensional geometry which can be either retrieved from measurements via high-resolution parameter estimation techniques 1.2.1, or input manually 1.2.2. The code for the former will be published in future versions of the IlmProp. The latter permits the generation of frequency selective time variant MIMO channels which display realistic correlation in time, frequency, and space. See 1.1.4 to better understand what are the aims of the IlmProp.

1.1.2 What is the current development status?

The public version of the IlmProp includes only a relatively small part of the features that have been already implemented. The main reason is that before publishing the code has to go through:

- cleaning up, optimization and proper commenting
- detailed and well organized documentation
- intensive debugging

These steps need a lot of time and therefore we start by publishing only a simpler yet reliable version with the most basic functionalities: a *beta*. Tables 1.1 and 1.2 shows the current versions available to download.

Table 1.1: Current development status of the **IlmProp**

Version	Date	Notes	Status
1.0.0	18 Dec. 2005	first release	BETA

Table 1.2: Current development status of the IlmProp **documentation**

Version	Date	Notes	Status
1.0.0	18 Dec. 2005	first release	BETA

1.1.3 How does it work?

The IlmProp relies entirely on a three-dimensional geometric representation of the environment and it has been designed to simulate a multi-user Multiple Input Multiple Output (MIMO) scenario, i.e., there is one Base Station (BS) and an arbitrary number of mobiles (M), where each can employ an antenna array, also referred to as smart antenna. An example of such a geometry can be seen in Figure 1.1.

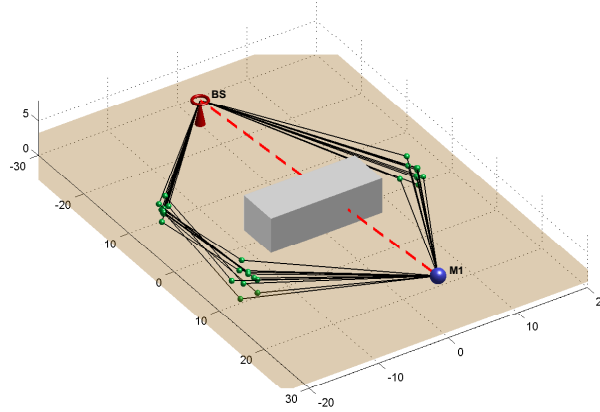


Figure 1.1: A simple IlmProp geometry.

Note that the IlmProp assumes an uplink channel. Therefore, often the base station is referred to as Rx and the mobiles as Tx . This is only a matter of convenience since the reciprocity of the wireless link allows us to use the computed channels for the downlink as well. The multi-path components are obtained by point-like scatterers, which can be placed at will. The model supports both single- and multiple-reflections. For the latter, it is necessary to input thorough which scatterers and in which order the paths propagate. The information about the scenario is stored in form of Cartesian coordinates and their evolution in time. The scenario may include obstacles (such as buildings), which can obstruct the propagation paths.

1.1.4 What does it NOT do?

The IlmProp is neither a system level nor a deployment channel model. In other words it is not a feasible solution to simulate large scale system level simulations for two reasons: (1) the large computational complexity, (2) its inability to generate realistic channels fully automatic. It is neither a deployment channel model because it is the user who has to define the propagation paths (by placing the scatterers). Therefore, the IlmProp is incapable of predicting the propagation conditions given a certain scenario (i.e., the position of obstacles).

The IlmProp and system level channel models

A system level channel model, such as the 3GPP SCM or the more recent WINNER WIMI channel model, is able, given very few parameters, to generate realistic channels which can be used for a large scale system level simulation. The parameters needed include the speed of the mobiles and the kind of scenario in which the users are (rural, bad urban, etc.). An exact knowledge of the environment or of the trajectories of the mobiles is not required. The main goal of a system level channel model is its speed in the computation of realistic channel impulse responses, since they are needed for large scale simulations, meaning for a number of mobile stations in the order of hundreds, and for a considerable number of time snapshots (several thousands). The parameters of the different Multi-Path Components (MPC) are generated from statistical distributions independently for each link, according to the specified scenario.

In the IlmProp, all mobile stations coexist in the same space, meaning that they potentially see the same clusters, or scatterers. For this reason it is not practical to generate more than a few dozens mobiles. Thus, only small-scale system level simulations are possible. The advantage of this approach is that the mobile will be realistically spatial correlated. Furthermore, the correlation will evolve with time implicitly as the mobiles move.

The IlmProp and deployment channel models

Deployment channel models are tools which predict the propagation conditions found in a specific environment. They follow two different approaches: full-wave and ray-based. The full-wave models solve directly the Maxwell equations, while the ray-based ones do it approximating the wave with a sum of rays. Both require a precise description of the environment, including information about the electromagnetic properties of the interacting objects in it contained.

The IlmProp uses MPC modelled as rays as well, but their propagation trajectory is set by the user, instead of being automatically computed. With this approach no detailed information about the environment modelled is needed, but at the same time there is no guarantee that the generated channel will be realistic at all.

1.2 Typical Applications

There are two major fields of applicability: measurement-based and non measurement-based.

1.2.1 Measurement-based applications

The IlmProp can be fed almost directly with the parameters obtained from a high-resolution parameter estimation technique such as ESPRIT or SAGE. The IlmProp geometry will mimic closely the channel measurement's environment and the scatterers are put in space properly so that the Directions of Departure (DoD), Directions of Arrival (DoA), Time

Delay of Arrival (TDoA), and path-stength obtained for the different paths match the results of the parameter estimations. This allows for a physical interpretation of the results, namely identifying the physical objects involved in the propagation of the paths extracted. The theory involved in this step will be published in An example of this application will be published soon.

1.2.2 non Measurement-based applications

The IlmProp is very useful in testing any algorithm which is based on the knowledge of the channel matrix and is sensible to frequency selectivness, time variance, and spatial correlation. Other features of the channel can be indirectly achieved by specific characteristics of the scenario modeled. Table 1.3 lists a few examples

Table 1.3: Channel features and the involved aspects of the scenarios

Channel	Scenario
frequency selectiveness	it is indirectly determined by the number and intensity of the echoes. A larger number of clusters at different Time Delay of Arrival (TDoA) provide for greater frequency selectiveness
time variance	greater speeds of the mobiles, changing reflection coefficients, or even moving scatterers increase the time variance
spatial correlation	the relative distance between two mobiles affects their spatial correlation. The spatial correlation between the antennas of the same mobile is determined by the number and angle distribution of the different propagation paths.
Rician K factor	the strenght of the reflection coefficients, as well as the number of scatterers determines the amount of power carried by Non Line of Sight (NLOS) components, thus affecting the K factor.
Diversity	the number of MPC affects the spatial diversity.
Eigenvalue distribution	the number of paths as well as their relative power affects the eigenvalue distribution of the channel matrix. The stronger the NLOS paths are, the richer, i.e., more white, will be the spatial correlation matrix.
capacity	in general the amount of power received directly affects the ergodic capacity. For a MIMO channel, given a fixed SNR the richer the scattering the higher the capacity.

Once the features of interest are decided, the user can go on in designing a scenario which presumably should give the desired channel characteristics. Once the channel is generated it is possible to retrieve the features, compare them with the desired ones, and proceed in a fine tuning of the geometry. For instance, if the channels of two mobiles display to little spatial correlation we can simply put them closer.

Examples of practical applications of the IlmProp can be found in the following publications:

- Block Diagonalization (BD) for SDMA [1]
- multihop concept [2]
- validation and investigation of an analytical model [3, 4]
- validation of a scheduling algorithm for SDMA (ProSched) [5]
- comparison of different multi user SDMA modulation schemes [6]
- validation of a clustering algorithm [7]

1.3 Requirements

The IlmProp is mostly written for MATLABTM 6.5 Rel. 13. The most computationally intensive routines are written in C. The IlmProp is available for Windows and Linux machines. Probably we will release a version for Macintosh in the future.

1.4 General License Agreement and Lack of Warranty

- The whole IlmProp (besides the routines `intersects` and `paths_getlencoeff_c`) is licensed under the GNU General Public License. Basically, you can use it for any purpose, provided that any programs or modifications you make and distribute are also licensed under the GNU GPL.
- This software is made available AS IS in the hope that it will be useful but WITHOUT ANY WARRANTY. The authors do not accept responsibility to anyone for the consequences of using it or for whether it serves any particular purpose or works at all. No warranty is made about the software or its performance.
- The receiving party agrees to acknowledge TUI's parenthood by referencing in every publication it may produce with the use of these software the IlmProp homepage - www.tu-ilmenau.de/ilmprop.
- The receiving party agrees to acknowledge the authors in every publication it may produce in the future based on the use of these packages.
- In order to ensure that any enhancement might benefit to the whole community using the software, the receiving party agrees to notify the authors of any change and/or improvement of the source code, and to document it.

1.5 Future Updates

Table 1.4 lists features and improvements of the IlmProp which will be made public in future versions:

Table 1.4: Future updates

User Interfaces
Several GUIs to analyze the generated channel. For instance to compute correlation, eigenvalues, cappacity etc.
Antenna Arrays
Sevaral measured beam patterns
The possibility to input your own beam patterns
Geometries
Sevaral <i>geometries</i> , many of which derived from measurements
Channel Generation
spherical wavefronts
automatic positioning of a cluster to simulate the ground reflection
full 3-D polarization
documentation on how to initialize a <i>geometry</i> from the prompt without GUIs
Diffuse Multipath Component

1.6 Credits

The original concept and development are by Giovanni Del Galdo, who remains curator of the IlmProp Project. Jörg Lotze joined later giving significant contributions to its development. The authors would like to thank the following people for numerous discussions, feedbacks and suggestions about the IlmProp:

- Prof. Martin Haardt (CRL - TUI)
- Prof. Reiner Thomä (EMT - TUI)
- Nicolai Czink (TUW, FTW)
- Hassan El-sallabi (TKK)
- Marko Hennöfer (CRL - TUI)
- Marko Milojevic (CRL - TUI)
- Veljko Stankovic (CRL - TUI)
- Martin Fuchs (CRL - TUI)

- Gerd Sommerkorn (EMT - TUI)
- Markus Landmann (EMT - TUI)
- Christian Schneider (EMT - TUI)

We hope that after the public release the list will grow with many beta testers.

Chapter 2

Getting Started

2.1 Downloading and Installing IlmProp

The IlmProp MATLAB code can be obtained at the IlmProp homepage:

<http://tu-ilmenau.de/ilmprop>

A direct link to the download page can be selected directly in the ‘Help’ menu of the main IlmProp GUI.

2.1.1 Installing

Simply unpack the IlmProp files (keeping the subdirectory structure) to any directory of your choice. In the MATLAB prompt go to the installation directory and call `ilmprop`. You do not need to add any directory to the PATH of MATLAB.

2.2 Quick Walkthroughs

Follow these tutorials to learn how to operate the IlmProp:

- Generate a new geometry and visualize it
- Generate the channel matrix, visualize it, and save it
- Modify an existing geometry
- Making a brand new geometry

2.2.1 Generate a new geometry and visualize it

From the main IlmProp GUI 6.1 you first need to generate a *geometry*.

Click on ‘NEW’ and on the next GUI 6.2 click on ‘Ok’.

You have just generated your first IlmProp *geometry*. This ran four scripts in the *geometry_scripts* subdirectory: `IPG_template1_G.m` which sets the general parameters such as the sampling grids in time and frequency and the center frequency, `IPG_template1_Rx.m` which contains the information about the receiver, `IPG_template1_Tx.m` defining the transmitter, and `IPG_template1_Env.m` setting the scatterers and obstacles. After a generation of one *geometry* a file `IPG_IPG_geometry_name_Info.mat` is created. It contains the information about the *geometry* which is displayed by the Geometry Selection GUI.

Click on ‘NEW’. The GUI now displays the parameters and a preview thumbnail of the geometry just created.

Click on ‘Cancel’ to go back to the main GUI.

Click on ‘GEOM’ to plot the 3D geometry 6.3. Here you can visualize the orientation of the antenna arrays by **clicking on the ‘Array Orientation’ checkbox**. Try other features of this GUI by clicking on the different checkboxes and buttons. A ToolTip will

be displayed if you leave the mouse pointer over a control. By **clicking on the ‘Export Figure’** you will open a new figure without any control (checkboxes, buttons and so on).

Go to the Main GUI and click on ‘ARRAY’ to visualize the Array Explorer 6.4. Experiment with the different plots.

2.2.2 Generate the channel matrix, visualize it, and save it

Let us now generate a channel. Note that it might take around 20 minutes on older PC’s.

Go to the Main GUI and generate the channel matrix by clicking ‘Calculate channel in total’ in the ‘Calculations’ menu. The `IlmProp` can compute the channel ‘in total’, or by storing the different components separately. In the latter case, the Line of Sight (LOS) and non Line of Sight (NLOS) will be stored in `data.H_LOS` and `data.H_NLOS`, respectively. In future versions it will be possible to compute also the Diffuse Multipath Component (DMC) and store it separately in `data.H_DMC`. This allows the user to sum the different components afterwards, multiplying them with a arbitrary factors to obtained a desired Rician K -factor.

Go to the Main GUI and click on ‘BELLO’ to visualize the Bello Explorer 6.5.

To save the computed channel matrix **go in the Main GUI and and click on ‘SAVE’** and select a location and a filename for the `.mat` file which will contain the *geometry* and computed channel matrix.

2.2.3 Modify an existing geometry

Let us now see how to modify an existing *geometry*. To do so, we need to modify the generating scripts and then initialize the *geometry*.

Click on ‘NEW’ and on the next GUI 6.2 click on ‘Ok’.

Click again on ‘NEW’ and on the next GUI 6.2 click on ‘All’.

The `IlmProp` calls the MATLAB editor and opens the four files needed for the generation of that geometry. You can now modify any parameter that you please, such as the array geometries, the position of clusters, Rx, and Tx, the reflection coefficients and so on.

Save your changes and go back to the Geometry Selection GUI.

Click on ‘Update’. Note that on the top right the GUI now informs that the parameters shown are not up to date. In fact they are the ones corresponding to the last initialization of this *geometry*.

Click on ‘Ok’ to generate the *geometry* with the modified parameters.

If you want you can keep modifying the scripts and re-generate the *geometry* by **clicking on ‘RE-RUN’** on the Main GUI 6.1. Remember that the changes in the scripts will affect the current *geometry* only when it is initialized, either by the Selection GUI 6.2 or by the ‘RE-RUN’ button on the Main GUI 6.1.

2.2.4 Making a brand new geometry

To generate a brand new geometry we suggest always to start from an existing one.

Click on ‘NEW’ and on the next GUI 6.2 select a *geometry* and click on ‘Copy’ and enter a new filename.

The IlmProp copies the four initialization files to new files. These files will be loaded in the MATLAB editor for you. Remeber to change in `IPG_filename_G.m` the `Geometry.name` and `Geometry.notes` fields. When you are done editing the new *geometry* **click on ‘NEW’ and on the next GUI 6.2 select it and click on ‘Ok’** to initialize it.

2.3 Templates

The first version of the IlmProp comes with two exemplary *geometries*. The first one, called `template1` is a single user scenario with 7 clusters and one obstacle. The latter provides for shadow fading as the mobile moves.

The second geometry is called `template2` and is a 3 user scenario. The topology of the *geometry* does not represent a realistic scenario and is just an example of the 3-D modelling capabilities of the IlmProp.

Both templates can be used as a basis for new *geometries*. Look directly in the code for more details.

Chapter 3

Frequently Used Terms

In the IlmProp code and GUIs, as well as throughout this document we use many terms which possess a specific meaning. This chapter presents a list of the most important ones, which will be helpful in avoiding misunderstandings while using the IlmProp.

array, antenna array, or smart antenna

The ‘array’, or ‘antenna array’ or ‘smart antenna’ consist of one or more sensors whose relative position and orientation is fixed.

antenna, sensor, or array element

With ‘antenna’, or ‘sensor’ or ‘array element’ we denote one of the elements constituting the antenna array.

channel

With channel we usually denote multi-dimensional array which stores the channel impulse response or the equivalent transform function in the time and frequency domains.

geometry

A *geometry* is the 3-D IlmProp environment from which the Channel Impulse Responses (CIR) are computed. It includes a Base Station (Rx) and one or more moving Mobiles (Tx). An arbitrary number of propagation paths link the Tx stations to the Rx. Obstacles, when defined, may obstruct the paths. The struct **Geometry** is the IlmProp data structure containing all the information needed to define a *geometry*.

up-vector

When defining the orientation of an antenna array, or an array sensor we need to specify a direction towards which the array or antenna is looking. The up-vector defines the direction of the local z axis, in other words, what direction the antenna or array would call ‘up’. For the up-vector of the camera look in the help of MATLABTM under **axes**.

Chapter 4

Coordinate Systems

The `IlmProp` defines scatterers, antennas and obstacles in Cartesian coordinates. It uses three different coordinate systems, as depicted in Figure 4.1

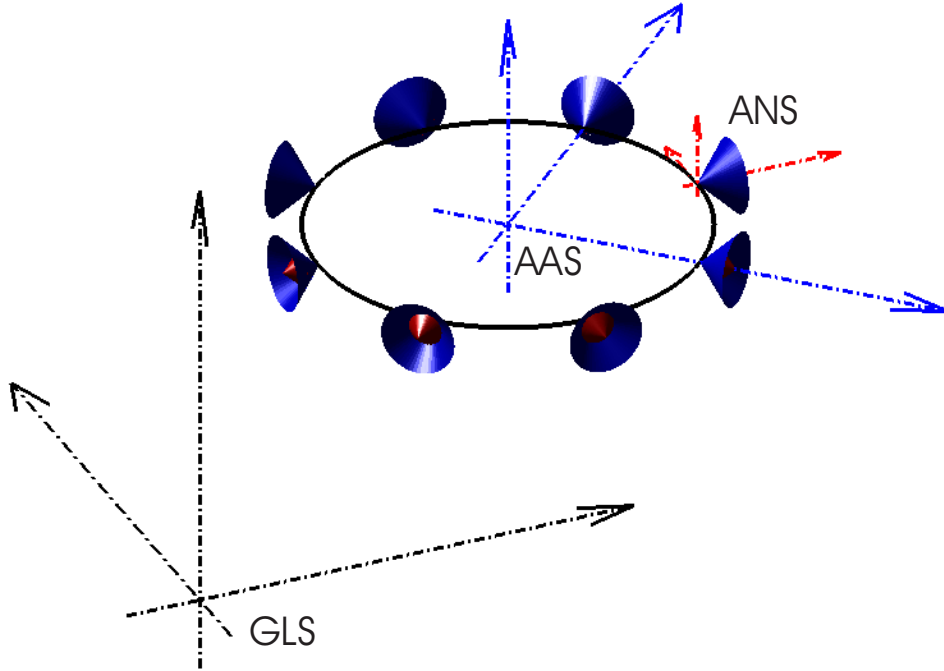


Figure 4.1: The three Cartesian coordinate systems used in the `IlmProp`. GLobal System (GLS), Antenna Array System (AAS), and ANtenna System (ANS).

GLobal System (GLS)

This is the main coordinate system. All positions and orientations of scatterers, Rx and Tx arrays, and obstacles are defined in this system.

Antenna Array System (AAS)

This is the coordinate system used to define the relative position and orientation of the elements of the antenna arrays. There exist one AAS for each antenna array defined in the *geometry*.

ANtenna System (ANS)

This is the coordinate system used to define the beam pattern of the array element. There exist one ANS for each sensor defined in the *geometry*.

Chapter 5

Data Structures

5.1 The struct Geometry

The struct `Geometry` contains the information about a 3-D *geometry*. Table 5.1 lists its fields. Note that `Geometry` must be a *global* variable in the Workspace. Use the Geometry Explorer 6.3 to visualize it.

Field	Type	Description
name	string	name of the <i>geometry</i> - NOT the filename
notes	string	user defined notes about the <i>geometry</i>
T	uint8	time snapshots in samples
time	[1× T double]	time axis in seconds
f0	double	center frequency in Hz
F	uint8	frequency bins in samples
freq	[1× F double]	frequency axis in Hz
date	string	string containing the date of generation
lambda0	double	center wavelength
Rx	[1×1 struct]	struct defining the receiver. See 5.1.1
Tx	[1× Users struct]	struct defining the transmitter. See 5.1.1
los_coefficients	[Users × T double]	Contains the LOS coefficients. See 5.1.2
Users	uint8	Number of Tx stations
S	struct	struct defining the scatterers. See 5.1.3
Obstacles	[1×1 struct]	struct defining the obstacles. See 5.1.5
paths	[K×N uint8]	struct defining the paths. See 5.1.4
generation_time	string	contains the duration of the generation process in seconds

Table 5.1: Fields contained in the struct `Geometry`

5.1.1 Geometry.Rx and Geometry.Tx

The struct `Geometry.Rx` contains the information about the array at the receiver. Its structure is identical to the one contained in `Geometry.Tx`, with the exception that `Geometry.Tx` can contain more than one array. The i -th array is accessed by

`Geometry.Tx(i)`.

Use the Array Explorer 6.4 to visualize it. Table 5.2 lists its fields.

Field	Type	Description
G	struct	struct containing the information about the array
array_rotx	$1 \times \text{Geometry.T}$	contains the angle for the rotation about the x axis in GLS
array_roty	$1 \times \text{Geometry.T}$	contains the angle for the rotation about the y axis in GLS
array_rotz	$1 \times \text{Geometry.T}$	contains the angle for the rotation about the z axis in GLS
positionc	$3 \times \text{Geometry.T}$	contains the coordinates in x , y , and z , of the center of the array for all time snapshots
O	uint8	number of antennas
Trans_GLS2AAS	$4 \times 4 \times \text{Geometry.T}$	transformation matrices to convert GLS coordinates into AAS
Trans_AAS2GLS	$4 \times 4 \times \text{Geometry.T}$	transformation matrices to convert AAS coordinates into GLS
Trans_ANS2AAS	$4 \times 4 \times \text{Geometry.T}$	transformation matrices to convert ANS coordinates into AAS
Trans_AAS2ANS	$4 \times 4 \times \text{Geometry.T}$	transformation matrices to convert AAS coordinates into ANS

Table 5.2: Fields contained in the struct **Geometry.Rx** and **Geometry.Tx**

Geoemtry.Rx.G

The struct **Geometry.Rx.G** or **Geometry.Tx.G** contains information about the antennas in the array. Use the Array Explorer 6.4 to visualize it. The relative distance and orientation of the sensors are assumed fixed. Table 5.3 describes its fields.

Field	Type	Description
M	uint8	number of antennas
positions	$3 \times M$	coordinates in x , y , and z of the antennas in AAS
antenna_rotx	$1 \times M$	contains the angles for the rotations about the x axis in AAS for each antenna
antenna_roty	$1 \times M$	contains the angles for the rotations about the y axis in AAS for each antenna
antenna_rotz	$1 \times M$	contains the angles for the rotations about the z axis in AAS for each antenna
azimuth	$1 \times K_{az}$	contains the azimuth axis for the Beampattern
elevation	$1 \times K_{az}$	contains the elevation axis for the Beampattern
Beampattern	$K_{el} \times K_{az}$	contains the matrix describing the beam pattern for the angles defined in azimuth and elevation . Note, it is not used in the Ilm-Prop. See 5.3
EADF	struct	struct containing the EADF. See 5.3
Type	string	string defining the antenna type. For instance ‘ULA’, ‘UCA’, ‘URA’, etc.
AntennaType	string	string defining the directivity of the antennas. For instance ‘vertical_dipole’, ‘omnidirectional’, etc. See <code>make_array.m</code>

Table 5.3: Fields contained in the struct `Geometry.Rx.G` and `Geometry.Tx.G`

Geoemtry.Rx.array_rot...

Three vectors describe the orientation of the M array sensors: `antenna_rotx`, `antenna_roty`, and `antenna_rotz`. They all have size $1 \times M$ and define the angles in radians by which the sensor is rotated about the x , y , and z axes, respectively. The i -th sensor, pointing towards the positive x axis and having up-vector towards the positive z axis, is first rotated about the x axis by `antenna_rotx(i)` radians. Similarly is rotated about y , and finally about z .

Geoemtry.Rx.Trans...

The matrices `Trans_GLS2AAS`, `Trans_AAS2GLS`, `Trans_ANS2AAS`, and `Trans_AAS2ANS` permit us to transform between different coordinate systems. See the functions `trans_AAS2ANS`, `trans_AAS2GLS`, `trans_ANS2AAS`, and `trans_GLS2AAS` for more details.

5.1.2 Geometry.los_coefficients

The `IlmProp` computes the path-loss for the Line Of Sight (LOS) component physically, considering the directivity function and the distance between Rx and Tx. However, for certain applications it is desirable to modify the power of the LOS. The `Geometry.los_coefficients` is a matrix $[\text{Geometry.Users} \times \text{Geometry.T}]$ which contains the factors for the different time snapshots and users, by which the power of the LOS component is multiplied. Setting `Geometry.los_coefficients` to all ones makes the `IlmProp` compute the path-loss following the physical laws. Setting a coefficient to 2, for instance, would double the power of the LOS for that particular User and time instant.

5.1.3 Geometry.S

The struct `Geometry.S` contains the information about the scatterers, namely where they located and their reflection coefficient.

The fields are: `Geometry.S.positions`, `Geometry.S.coefficients`, and `Geometry.S.O`. The field `Geometry.S.positions` is a matrix of size $3 \times \text{Geometry.T} \times \text{Geometry.S.O}$. It contains the coordinates x , y , and z , in GLobal System (GLS), of the position of `Geometry.S.O` scatterers for all `Geometry.T` time snapshots.

The matrix `Geometry.S.coefficients` has size $\text{Geometry.S.O} \times \text{Geometry.T}$ and contains the complex reflection coefficients of the `Geometry.S.O` scatterers for each of the `Geometry.T` time snapshots.

5.1.4 Geometry.Obstacles

This field defines the paths connecting the Tx to the Rx through the scatterers. It is a matrix of size $K \times N_{\max}$. The i -th row is defined as:

$$\begin{bmatrix} \text{User} & S_1 & S_2 & \dots & S_{N,i} & 0 & \dots & 0 \end{bmatrix} \quad (5.1)$$

where User is either the the number of the Tx station, thus from 1 to `Geometry.Users`, or 0 so that all Tx stations have this path. The terms S_1 to $S_{N,i}$ are the indices of the scatterers through which the path propagates. From $S_{N,i}$ the path reaches the receiver. The remaining zeros in the row are to reach the size of the matrix.

Example: the row

$$\begin{bmatrix} 0 & 3 & 5 & 0 \end{bmatrix} \quad (5.2)$$

specifies a path starting from all Tx stations, reaching the Receiver with a double bounce through the third and fifth scatterers. If not specified in the `IPG_template1_Env.m` file, the `IlmProp` takes by default all single reflection paths.

5.1.5 Geometry.Obstacles

The struct `Geometry.Obstacles` defines the obstacles in the *geometry*. If no obstacles are present you can avoid initializing it. The fields are: `Geometry.Obstacles.x`, `Geometry.Obstacles.y`, and `Geometry.Obstacles.z`. Note that the obstacles can only be boxes oriented as the GLS axes. Each field is a matrix of size $2 \times K$, where K is the number of obstacles. For each column the matrices define the extremes of the box in the x , y , and z axes. The field `Geometry.Obstacles.O` contains the number of obstacles present. It is automatically generated by the `IlmProp` when initializing a *geometry* 6.2.

5.2 The struct data

The struct `data` contains the channel matrix as well as other outputs of the `IlmProp`.

Table 5.4 lists its fields. Note that `data` must be a *global* variable in the Workspace.

Field	Type	Description
H	$[M \times (N \cdot K) \times T \times F \text{ double}]$	channel matrix, total, see 5.2.1
H_DMC	$[M \times (N \cdot K) \times T \times F \text{ double}]$	channel matrix, Diffuse Multipath Component, see 5.2.1
H_LOS	$[M \times (N \cdot K) \times T \times F \text{ double}]$	channel matrix, Line Of Sight, see 5.2.1
H_NLOS	$[M \times (N \cdot K) \times T \times F \text{ double}]$	channel matrix, Non Line Of Sight, see 5.2.1

Table 5.4: Fields contained in the struct `data`

5.2.1 data.H, data.H_DMC, data.H_LOS, and data.H_NLOS

These matrices contain the channel coefficients for the the M receive antennas, $(N \cdot K)$ transmit antennas, T time snapshots and F frequency bins. Note that the $(N \cdot K)$ transmit antennas (for the mobiles) are stacked. All K mobiles possess the same number of array sensors N . The time axis in seconds can be found in `Geometry.time`, while the frequency axis in Hertz is in `Geometry.freq`. For the base-band representation of the frequency axis use `Geometry.freq - Geometry.f0`. When computed, `data.H` contains the total channel

transfer matrix, while `data.H_LOS` and `data.H_NLOS` contain the Line Of Sight (LOS) and Non-Line Of Sight (NLOS) parts separately. The matrix `data.H_DMC` contains the Diffuse Multipath Component (DMC). This feature will be implemented in future versions of the `IlmProp`.

5.3 The EADF

The `IlmProp` features the Effective Aperture Distribution Function. It is an efficient representation of the polarimetric antenna response. This reduced description permits us to efficiently interpolate the beam pattern to gather the antenna response for an arbitrary direction in azimuth and elevation, rendering the EADF especially suited for Geometry-based channel modelling. Moreover, the EADF provides a continuous description of the array manifold and its derivatives with respect to azimuth and elevation. The latter is valuable for the performance evaluation of an antenna array as well as for gradient-based parameter estimation techniques.

Future versions of this document will describe in detail the EADF. For now, please refer to the publication [8].

Chapter 6

The Graphical User Interfaces

6.1 The main IlmProp GUI

The main IlmProp GUI controls all functionalities of the IlmProp. It is simply called from the prompt by the command `ilmprop`. The top buttons 1 to 7 call the most useful features. At the bottom the GUI displays with colors which data structures are present in the Workspace.

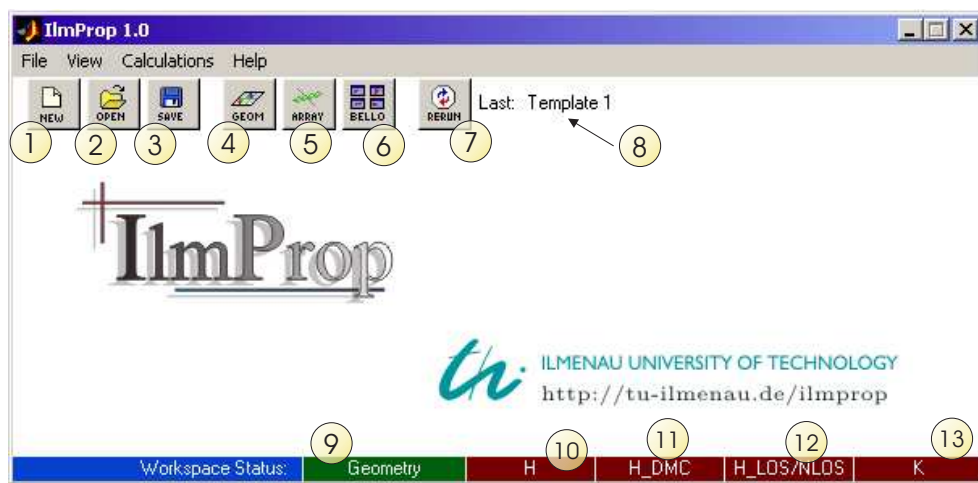


Figure 6.1: The Main IlmProp GUI

Number	Description
1	Opens the Geometry Selection GUI 6.2
2	Calls a dialog to load an existing <code>.mat</code> file containing a <i>geometry</i> and/or channel matrix
3	Saves the current Geometry and/or data in a <code>.mat</code> file
4	Opens the Geometry Explorer 6.3
5	Opens the Array Explorer 6.4
6	Opens the Bello Explorer 6.5
7-8	Regenerates the <i>geometry</i> which was last generated. Its name is displayed in 8
9	If green indicates that the struct Geometry is in the Workspace
10	If green indicates that the channel matrix is in the Workspace, saved as data.H
11	If green indicates that the Diffuse Multipath Component is in the Workspace, saved as data.H_DMC (only in future versions)
12	If green indicates that the Line Of Sight (LOS) and Non Line Of Sight (NLOS) components of the channel matrix are saved as LOS and NLOS components are in the Workspace, saved as data.H_LOS and data.H_NLOS , respectively
13	If green indicates that the Rician <i>K</i> factor has been calculated and save in data.K

Table 6.1: Description of the main IlmProp GUI elements

6.2 The Geometry Selection

This GUI allows the user to select which *geometry* to generate. Pre-existing *geometry* script files can be previewed. The graphical user interface is called by either clicking on the New icon in the main GUI (see section 6.1) or by the **New Geometry** menu item. It searches the directory **geometry_scripts** for all valid geometries and displays them in a popup menu. The searching algorithm is described below. The GUI displays a preview and lists the parameters only for those *geometries* which have been already generated. Figure 6.2 shows the GUI while table 6.2 explains its elements.

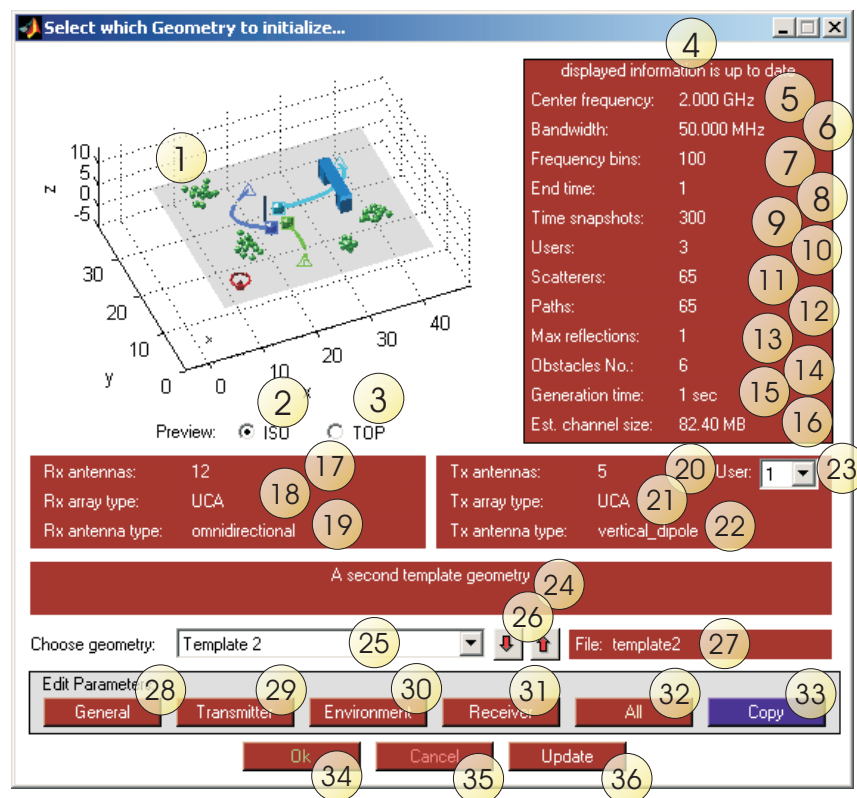


Figure 6.2: The Geometry Selection GUI

Number	Description
1	Preview of the geometry
2	Show the preview in a 3D isometric view (as in the example)
3	Show the preview from top
4	Indicates if the displayed information is up to date (no changes in script files after last initialisation)
5	Center frequency of the geometry
6	Bandwidth (difference of the maximum and minimum frequency)
7	Number of frequency samples
8	End time of the geometry in seconds
9	Number of time snapshots
10	Number of users
11	Number of scatterers in the environment
12	Number of paths (estimated, i.e. in a single user geometry this number is correct, multiple users might have some paths in common, which are counted only once.)
13	Maximum number of reflections in the environment (single-bounce, double-bounce, multi-bounce)
14	Number of Obstacles
15	Time it took to generate last time
16	Expected size of the channel in memory
17, 20	Number Rx and Tx antennas respectively
18, 21	Types of Rx and Tx antenna arrays respectively, e.g. ULA, URA, UCA, ...
19, 22	Types of Rx and Tx antennas respectively, e.g. dipole, omnidirectional, ...
23	Popup menu to choose the user for which to display antenna information
24	Notes regarding the geometry
25	Popup menu to choose between all existing geometries
26	Buttons to go one geometry up or down in the list
27	The name of the script files for geometry generation (center part)
28, 29, 30, 31	Buttons to edit the scripts for the corresponding part of the geometry (G/Rx/Tx/Env)
32	Button to open all generation scripts of the current geometry in the MATLAB editor
33	Button to copy the scripts of currently selected geometry to new files and open those in the MATLAB editor (i.e. creates a new geometry from the selected one)

Number	Description
34	Closes the GUI and generates the selected geometry
35	Closes the GUI without generating a new geometry
36	Update the geometry list (in the popup menu) and all the displayed information. This is useful if some files have been copied or edited after the GUI was opened.

Table 6.2: Description of the Geometry Selection GUI elements

The searching algorithm looks for *geometries* in the `geoemtry_scripts` directory, where four initialisation files exist. These files are `IPG_geometryname_G.m`, `IPG_geometryname_Tx.m`, `IPG_geometryname_Env.m`, and `IPG_geometryname_Rx.m`, where `geometryname` is the name of the geometry. It also gets name and notes from the `IPG_geometryname_G.m` file and displays them in the corresponding fields of the GUI. Furthermore it looks for a file called `IPG_geometryname_Info.mat` which contains information for the other fields of the GUI and the previews. This file is created automatically after each initialisation of a geometry. If it is older then the scripts, i.e. the scripts have been changed after the last initialisation, a warning is shown in the GUI field number 4 (see figure 6.2).

To edit the script files of the selected geometry, simply click on one of the buttons with numbers between 28 and 32 in the figure. The program will open the corresponding files in the MATLAB editor. After changes are made, click on the update button in the GUI (number 36) and the list in the popup menu as well as the displayed information will be updated. If you want to create a new *geometry* by changing an existing one, click on the ‘Copy’ button. A dialog will pop up to ask for the new filename. By clicking on ‘Ok’ the selected geometry will be copied to new files which will be opened in the MATLAB editor. After you made your changes click on the ‘Update’ button on the GUI to get the new geometry into the list.

Note that this GUI is waiting for a user response, i.e., you cannot use the MATLAB command window while the GUI is opened.

Known Bugs:

- On some machines, where the standard font is larger, the displayed information is longer then the text fields and therefore it gets truncated. The GUI was designed with MATLAB 6.5 on a Windows machine.

6.3 The Geometry Explorer

The Geometry Explorer GUI allows the user to visualize three-dimensionally the *geometry* defined in the struct `Geometry`. The GUI can be opened by the IlmProp main GUI 6.1 or by the prompt with the command `plot_geometry_gui`. The sidebar at the bottom allows to change the time instant currently plotted. The check boxed on the right give many visualization options. The buttons at the top toggle different view points. Figure 6.3 shows a screenshot of the GUI, while Table 6.3 explains in detail all the commands included.

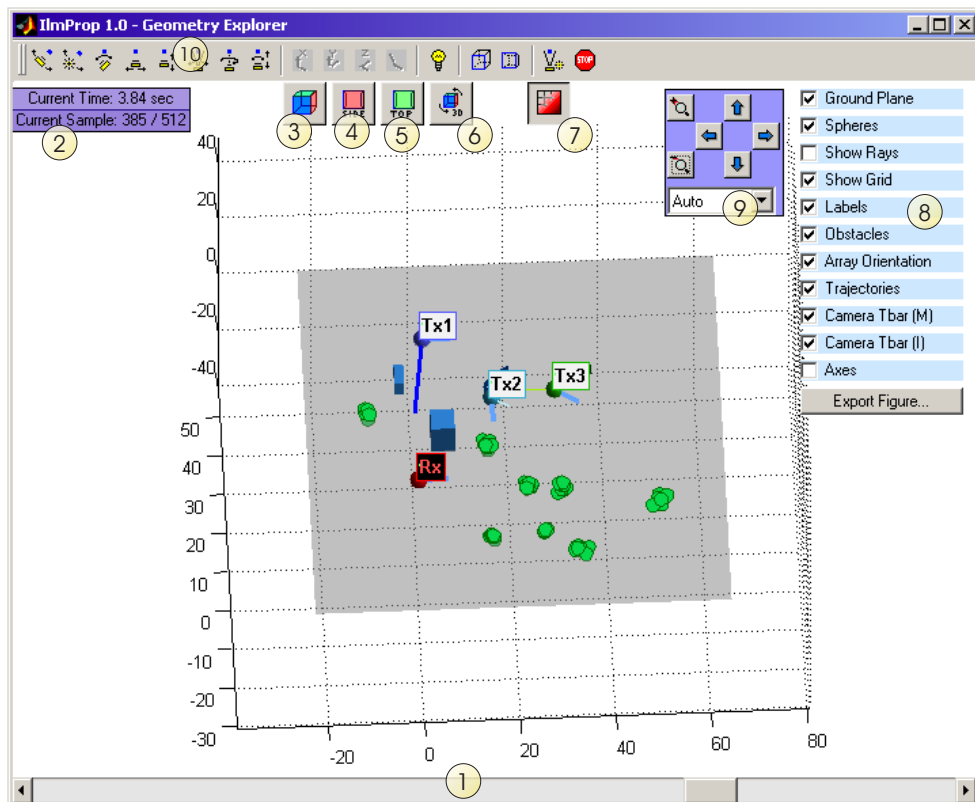


Figure 6.3: The Geometry Explorer GUI

Number	Description
1	The sidebar selects the time snapshot currently plotted
2	The text boxes display the current time snapshot, in seconds and in samples
3-5	the buttons control the point of view. The toolbars called by the checkboxes ‘Camera Tbar’ allow more possibilities
6	allows the user to rotate with the mouse the 3-D plot
7	toggles the 3-D renderer (OpenGL)
8	The checkboxes modify the plot. Over with the mouse on them to see a ToolTip
9	The IlmProp Camera ToolBar. The popup menu allows the user to focus on a specific Mobile or on the Base Station
10	The standard MATLAB Camera ToolBar

Table 6.3: Description of the Geometry Explorer GUI

6.4 The Array Explorer

The Array Explorer GUI allows the user to visualize three-dimensionally the antenna arrays defined in the struct **Geometry**. The GUI can be opened by the IlmProp main GUI 6.1 or by the prompt with the command `plot_array_gui`. The buttons at the top control the point of view and the rendering. The popup menu named ‘Plot’ selects what to be displayed. The available plots are:

- geometry 6.4.1
- directivity 6.4.2
- directivity [dB] 6.4.2
- all beams 6.4.3
- all beams [dB] 6.4.3
- directivity in az 6.4.4

Figures 6.4, 6.5, 6.6, and 6.7 show the GUI for different plots. Tables 6.4, 6.5, 6.6, and 6.7 describe their functionalities.

6.4.1 Array Explorer - geometry

This plot shows the position and orientations of the array sensors. The coordinate system displayed is the Antenna Array System (AAS).

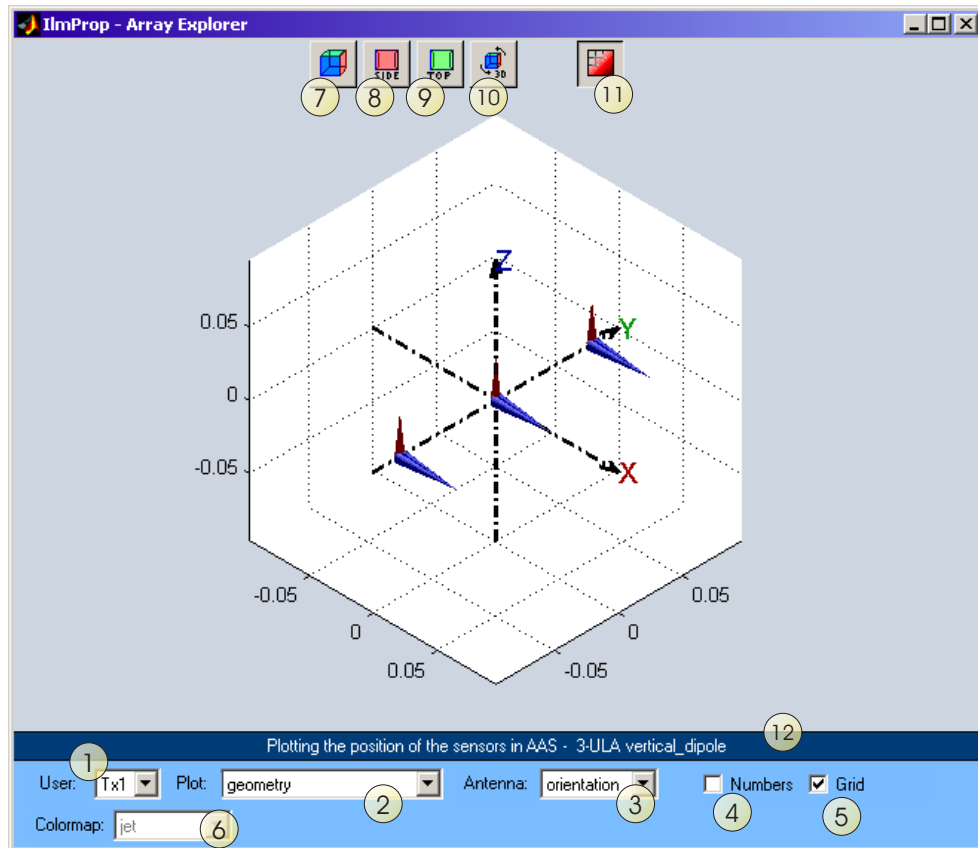


Figure 6.4: The Array Explorer GUI for the plot ‘geometry’.

Number	Description
1	selects the array at the Rx or at one of the transmitters
2	selects which plot to display
3	selects the graphical representation for the array sensors. With ‘orientation’ a blue cone shows the direction towards which the anntenna is oriented, while a red cone indicates the antenna’s <i>up vector</i> , i.e., its positive vertical axis
4	displays the numbering of the antennas
5	toggles the grid
6	selects the colormap
7-9	the buttons control the point of view
10	allows the user to rotate with the mouse the 3-D plot
11	toggles the 3-D renderer (OpenGL)
12	information about the antenna array displayed

Table 6.4: Description of the Array Explorer GUI elements for the plot ‘geometry’.

6.4.2 Array Explorer - directivity

This plot shows the directivity function for one antenna. The coordinate system displayed is the Antenna Array System (ANS).

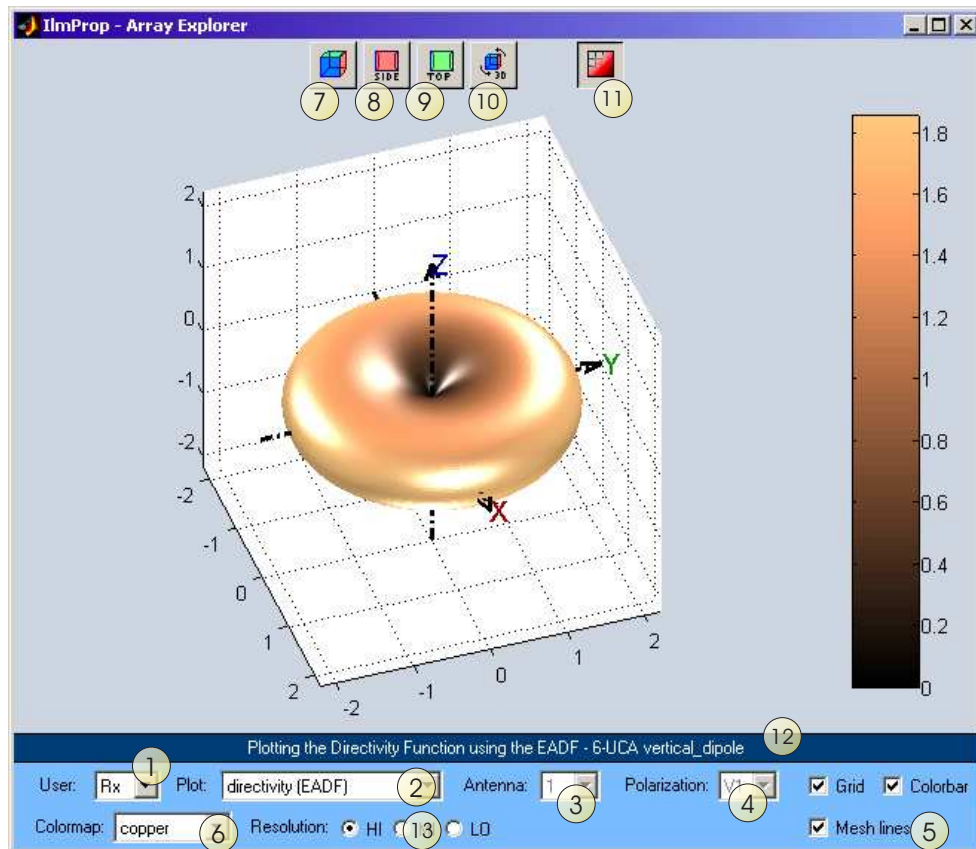


Figure 6.5: The Array Explorer GUI for the plot 'directivity'.

Number	Description
1	selects the array at the Rx or at one of the transmitters
2	selects which plot to display
3	selects which antenna to plot. If inactive all antennas have the same beam pattern
4	selects the polarization (future versions)
5	toggles the mesh grid. Works only when not rendered
6	selects the colormap
7-9	the buttons control the point of view
10	allows the user to rotate with the mouse the 3-D plot
11	toggles the 3-D renderer (OpenGL)
12	information about the antenna array displayed
13	selects the angle resolution with which to plot the directivity function.

Table 6.5: Description of the Array Explorer GUI elements for the plot ‘directivity’.

6.4.3 Array Explorer - all beams

This plot shows the directivity function of all antennas. The coordinate system displayed is the Antenna Array System (AAS). Note that the directivity functions are scaled so that the maximum is equal to half of the minimum distance between two elements. The plot is useful in comparing the directivity functions with respect to each other and to the position and orientation of the array elements.

Number	Description
1	selects the array at the Rx or at one of the transmitters
2	selects which plot to display
3	selects which antenna to plot. If inactive all antennas have the same beam pattern
4	selects the polarization (future versions only)
5	toggles the mesh grid. Works only when not rendered
6	selects the colormap
7-9	the buttons control the point of view
10	allows the user to rotate with the mouse the 3-D plot
11	toggles the 3-D renderer (OpenGL)
12	information about the antenna array displayed

Table 6.6: Description of the Array Explorer GUI elements for the plot ‘all beams’.

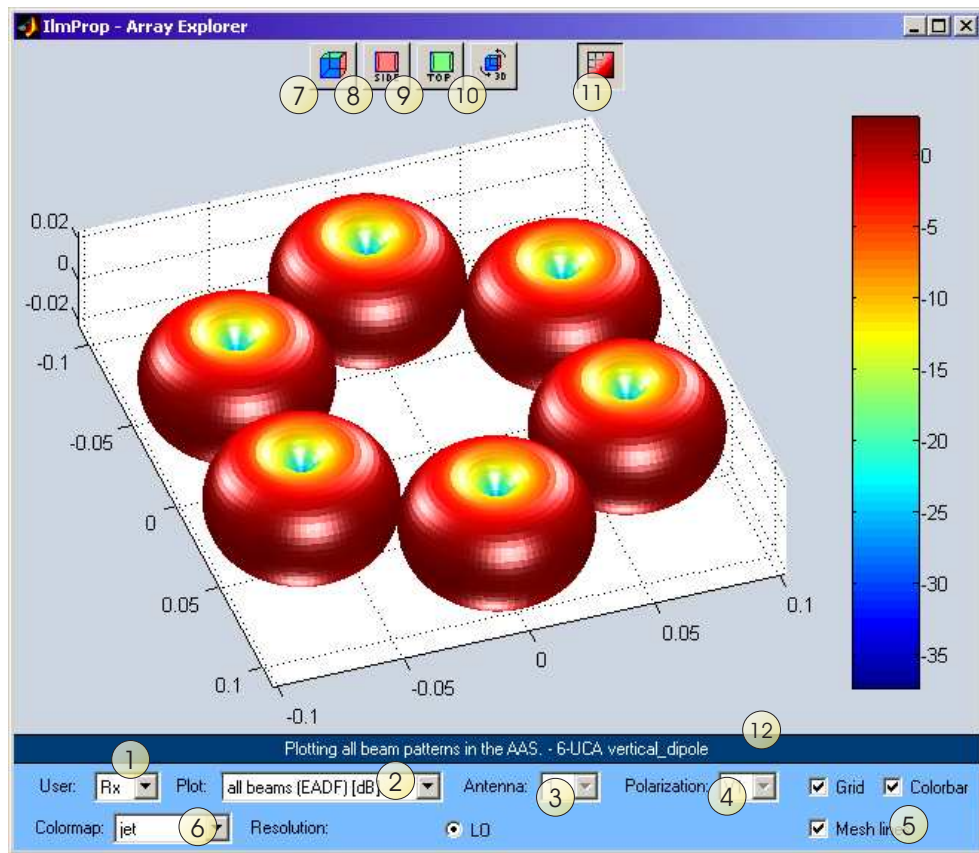


Figure 6.6: The Array Explorer GUI for the plot 'all beams'.

6.4.4 Array Explorer - directivity in az

This plot shows the directivity function of one antenna for zero elevation, i.e., at the equator. The coordinate system displayed is the Antenna Array System (ANS). The direction of maximum directivity is shown.

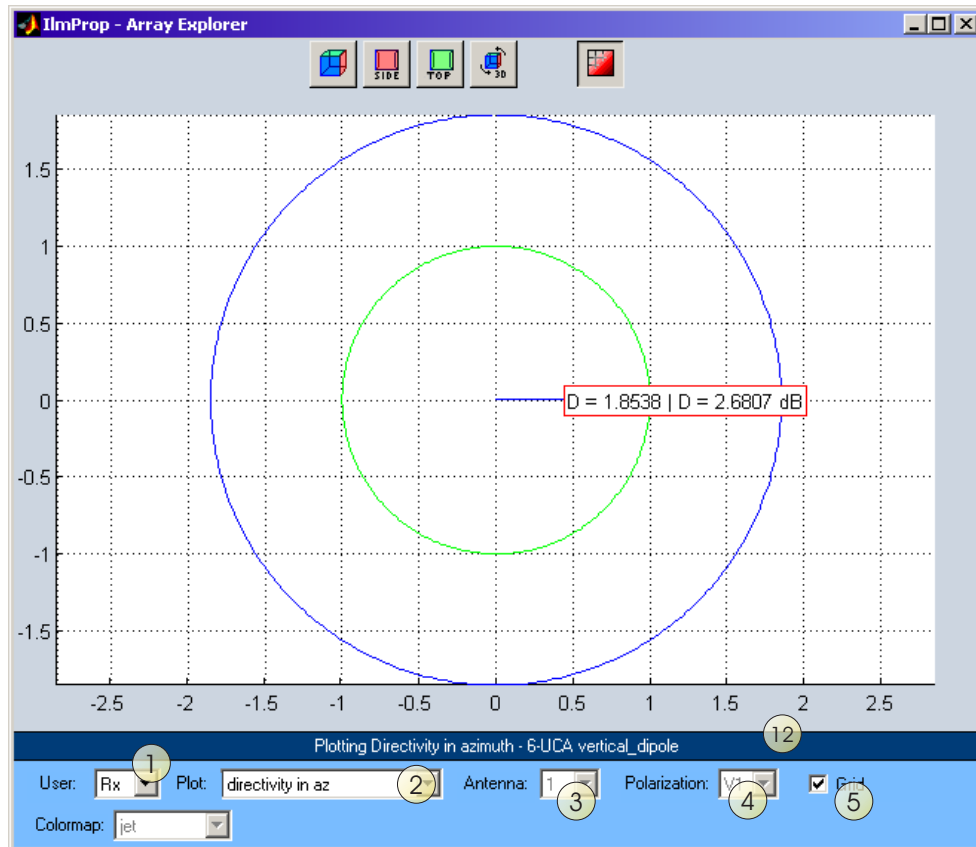


Figure 6.7: The Array Explorer GUI for the plot 'directivity in az'.

Number	Description
1	selects the array at the Rx or at one of the transmitters
2	selects which plot to display
3	selects which antenna to plot. If inactive all antennas have the same beam pattern
4	selects the polarization (future versions only)
5	toggles the grid

Table 6.7: Description of the Array Explorer GUI elements for the plot ‘directivity in az’.

6.5 The Bello Explorer

Using this graphical user interface you can look at the channel in the four Bello domains, i.e. the time variant channel impulse response $h(t, \tau)$, the time variant transfer function $H(t, f)$, the delay Doppler-spread function $h(f_D, \tau)$ and the frequency Doppler-spread function $(H(f_D, f))$. These functions are calculated via fast fourier transforms. The GUI gets called by either clicking on the Bello icon in the main GUI (see section 6.1) or by selecting the menu item **Bello Explorer**. Figure 6.8 shows the GUI while table 6.8 explains its elements.

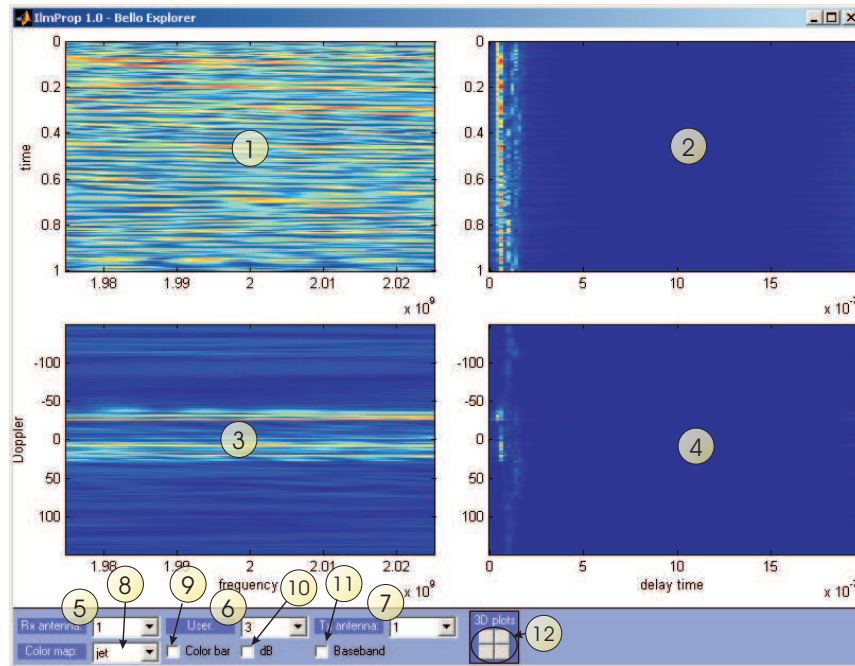


Figure 6.8: The Bello Explorer

If the Bello Explorer is opened when the channel has been calculated before (i.e. it is present in the data structure), it takes this channel, calculates the transformations and displays it. If only the **Geometry** was initialised, the Bello Explorer asks if it should

Number	Description
1	Time-variant channel transfer function from top
2	Time-variant channel impulse response from top
3	Frequency Doppler-spread function from top
4	Delay Doppler-spread function from top
5	Popup menu to select the Rx antenna
6	Popup menu to select the user
7	Popup menu to select the Tx antenna
8	Popup menu to select the color map for the plots
9	Checkbox for plotting the color bar next to each of the plots
10	Checkbox to plot everything on dB scale ($20 \log_{10}(\cdot)$)
11	Checkbox to show the frequency axis in baseband
12	Buttons to open a new figure with the 3D plot of the corresponding Bello domain (with the current settings)

Table 6.8: Description of the Bello Explorer elements

calculate the parts of the channel, which are needed for display (for selected user and antennas). Depending on the *geometry* this might take several minutes.

If one of the buttons of number 12 is clicked (see figure 6.8), a new figure with a three dimensional plot of the corresponding domain is opened, i.e. for the upper left button the time-variant channel transfer function is shown in the new figure including the standard MATLAB toolbars and menus. Therefore the figure can be modified and saved.

Chapter 7

The Channel Generation

This chapter will explain in detail how the `IlmProp` computes the channel matrix from a given *geoentry*. Most of the information is already provided in the code. For this reason, in this first BETA version, we do not provide any details in this guide.

Bibliography

- [1] G. Del Galdo and M. Haardt, “Comparison of zero-forcing methods for downlink spatial multiplexing in realistic multi-user MIMO channels,” in *Proc. IEEE Vehicular Technology Conference 2004-Spring, Milan, Italy*, May 2004.
- [2] Marko Hennhöfer, Martin Haardt, and Giovanni Del Galdo, “Increasing the throughput in wireless multi-hop networks by using spatial multiplexing,” in *Proc. IEEE Vehicular Technology Conference (VTC’04 spring)*, Milano, Italy, May 2004.
- [3] G. Del Galdo, M. Milojevic, M. Haardt, and M. Hennhöfer, “Efficient channel modelling for frequency selective MIMO channels,” in *Proc. ITG Workshop on Smart Antennas, Munich*, Mar. 2004.
- [4] G. Del Galdo, M. Haardt, and M. Milojevic, “A subspace-based channel model for frequency selective time variant mimo channels,” in *Proc. 15th Int. Symp. on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Barcelona, Spain, 2004.
- [5] M. Fuchs, G. Del Galdo, and M. Haardt, “A novel tree-based scheduling algorithm for the downlink of multi-user MIMO systems with ZF beamforming,” Philadelphia, PA, March 2005, vol. 3, pp. 1121–1124.
- [6] Martin Haardt, Veljko Stankovic, and Giovanni Del Galdo, “Efficient multi-user MIMO downlink precoding schemes,” in *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP 2005)*, Puerto Vallarta, Mexico notes=accepted for publication, December 2005.
- [7] Nicolai Czink and Giovanni Del Galdo, “Validating a novel automatic cluster tracking algorithm on synthetic IlmProp time-variant mimo channels,” *COST 273, TD(05)105*, 9–11 November 2005, Lisbon, Portugal.
- [8] M. Landmann and G. Del Galdo, “Efficient antenna description for MIMO channel modelling and estimation,” *European Microwave Week*, 2004.