

GEMV²: Geometry-based Efficient propagation
Model for V2V communication

<http://vehicle2x.net>

User Manual, Version 1.2

Mate Boban
mate.boban@live.com
<http://mateboban.net>

Copyright © 2014-2015 Mate Boban

Contents

1	Introduction	1
2	Getting Started	2
2.1	Installation	2
2.2	Running the simulation	2
2.3	Simulation output	2
2.3.1	Visualizing the output of GEMV ²	3
2.4	GEMV ² function dependencies	3
2.4.1	External function dependencies	3
2.5	What is in the package?	7
3	Building and Foliage Outlines	7
3.1	Using Outlines of Buildings and Foliage from OpenStreetMap . .	7
3.2	Converting Shapefiles to OpenStreetMap	8
4	Vehicular Mobility and Outlines: SUMO Floating Car Data	9
5	Referencing GEMV²	9
	Acknowledgements	10
	References	10

1 Introduction

This user manual describes the MATLAB implementation of GEMV², a geometry-based, efficient propagation model for vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication [1]. The main features of GEMV² are:

- Uses outlines of vehicles, buildings, and foliage to distinguish the following three types of links: line of sight (LOS), non-LOS due to vehicles, and non-LOS due to static objects
- Deterministically calculates large-scale signal variations (path-loss and shadowing)
- Calculates the small-scale power variations using the number and size of objects around the communicating vehicles
- Models both V2V and V2I links
- Validated against V2V and V2I measurements in urban, suburban, highway, and open space environments
- Imports vehicular mobility from SUMO; uses floating car data format to generate vehicle outlines
- Imports outlines of buildings and foliage from OpenStreetMap
- Easily extensible to import vehicular mobility and object outlines from other data sources
- Exports the visualization of vehicular communication to KML format (for use with Google Earth or NASA World Wind)
- Implemented in MATLAB; free of charge and openly distributed

This user manual describes how to install and run GEMV², what type of output it generates, and gives a high-level overview of GEMV² code. For a detailed description of GEMV² from a research standpoint, please refer to the following paper:

Mate Boban, João Barros, and Ozan K. Tonguz: “Geometry-Based Vehicle-to-Vehicle Channel Modeling for Large-Scale Simulation,” IEEE Transactions on Vehicular Technology, Vol. 63, No. 9, November 2014, doi:10.1109/TVT.2014.2317803 [PDF][BIB]

MATLAB implementation of GEMV² can be used as a standalone tool for analyzing V2V and V2I communication in terms of key communications and networking performance metrics: received power, packet delivery rate, interference levels, neighborhood size, etc.). GEMV² can also be used as a propagation model for discrete-time network simulators (e.g., NS-3, JiST/SWANS/STRAW),

either offline (i.e., by providing the output from the MATLAB implementation to the simulator) or online (by re-implementing GEMV² in the simulator).

2 Getting Started

2.1 Installation

Installation of GEMV² is quite simple: download the `.zip` file containing GEMV², unzip to a directory, and you're ready to go! Since GEMV² code is organized in package folders¹, there is no need to add any directories to MATLAB path. Furthermore, care has been taken so that GEMV² depends on as few MATLAB toolboxes as possible. That said, mapping, statistics and machine learning toolbox can speed up the simulation considerably, since the “backup” functions, used in case these toolboxes are unavailable, are far slower. GEMV² should work on any platform running recent version of MATLAB. However, because it uses package folders, it will not work with MATLAB R2007b and older without some modifications. It was tested with MATLAB R2011a, R2011b, R2012b, and R2013a under Mac OS X and Windows.

2.2 Running the simulation

To run the simulation, go to GEMV² root directory and type `runSimulation` in MATLAB command window. This will kick off the simulation by loading the script `runSimulation.m`, which in turn loads `simSettings.m`, the script that contains the default settings, including the source files for vehicle, building, and foliage outlines, propagation-related parameters, visualization options, etc.

2.3 Simulation output

After the simulation is finished, relevant variables related to received power, small- and large-scale variations, link types, etc., are saved in two formats:

- MATLAB `.mat` file (suitable for easy manipulation in MATLAB);
- comma-separated values (`.csv` files), with each variable saved in a different file (suitable for analyzing the results outside MATLAB).

Output files can be found in `outputSim` directory, located in the main GEMV² directory. The file names are appended with current date; note that if you do

¹For details, please refer to http://www.mathworks.com/help/matlab/matlab_oop/scoping-classes-with-packages.html

not copy/rename the output files and if there were simulation runs previously that day, the output from those runs will be overwritten.

2.3.1 Visualizing the output of GEMV²

GEMV² generates the V2V and V2I simulation output in Keyhole Markup Language (KML) format. To visualize the result, you can use Google Earth² or NASA World Wind [2], though any tool for displaying three-dimensional virtual globe³ can be used, provided that it supports KML format.

Figure 1 shows the Google Earth visualization in terms of power, whereas Fig. 2 shows the visualization of neighborhood size (defined as the number of vehicles that the ego vehicle can communicate with directly/without multi-hop communication). If the simulation is composed of multiple time-steps, the output is an animation of all time-steps (a number of sample videos is available at <http://vehicle2x.net/videos>). The visualization enables easier understanding of propagation characteristics in vehicular communication. More details are available in the demo paper on visualization [3]. Note: if the output is large (large being a few hundred thousand communication pairs in total or, alternatively, a few hundred megabytes), the following can occur:

- generating the .kml file can take a long time;
- opening the Google Earth output can be cumbersome and could even lead to Google Earth throwing an “out of memory” error (e.g., for .kml files larger than a few hundred megabytes).

Therefore, make sure to turn the visualization off in such cases.

2.4 GEMV² function dependencies

Figure 3 shows GEMV² function dependencies. The functions have been organized in package folders based on their functionality (e.g., functions for loading data are in `+loadFunctions` directory, functions for plotting are in `+plotFunctions`, etc.). Each function shown in Fig. 3 has been thoroughly commented; therefore, for a detailed description, open the containing .m file.

2.4.1 External function dependencies

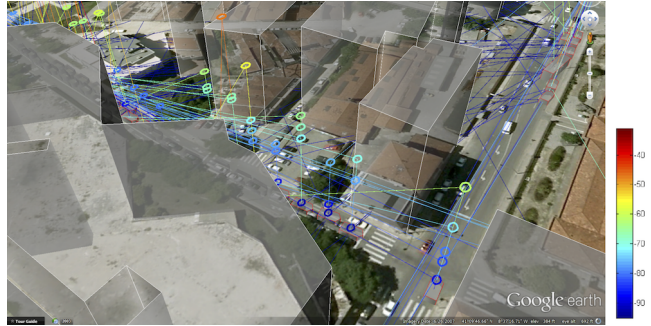
GEMV² requires the following functions available at MATLAB Central File Exchange: <http://www.mathworks.com/matlabcentral>.

²<http://www.google.com/earth/>

³http://en.wikipedia.org/wiki/Virtual_globe



(a) Snapshot showing a part of transmit-receive pairs in the city of Porto.



(b) Street-level view.

Figure 1: Visualization of received power generated by GEMV². Color bars are in dBm.

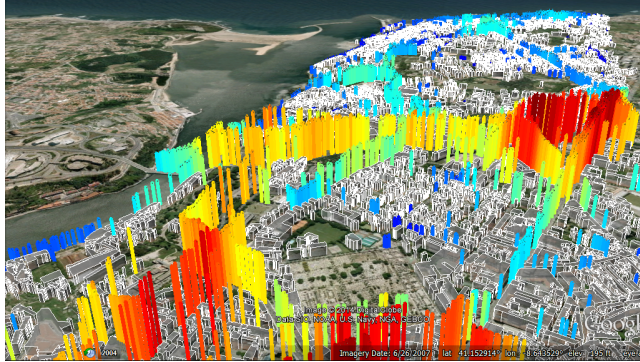
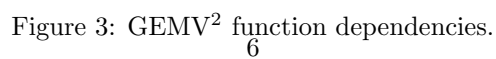


Figure 2: Visualization of neighborhood size generated by GEMV².

- deg2utm and utm2deg © 2006 by Rafael Palacios: <http://www.mathworks.com/matlabcentral/fileexchange/10915> and <http://www.mathworks.com/matlabcentral/fileexchange/10914>
- Line Simplification © 2009 by Wolfgang Schwanghart: <http://www.mathworks.com/matlabcentral/fileexchange/21132>
- Google Earth Toolbox © 2012 by Scott Lee Davis: <http://www.mathworks.com/matlabcentral/fileexchange/12954>
- Fast and Robust Curve Intersections © 2006 by Douglas Schwarz <http://www.mathworks.com/matlabcentral/fileexchange/11837>
- Line-Line Intersection (2d) © 2012 by Sebastian W: <http://www.mathworks.com/matlabcentral/fileexchange/35606>
- OpenStreetMap Functions © 2012 by Ioannis Filippidis: <http://www.mathworks.com/matlabcentral/fileexchange/35819>
- Fast Range Search through JIT (ver 2) © 2009 by Yi Cao: <http://www.mathworks.com/matlabcentral/fileexchange/19480>
- xml2struct © 2010 by Wouter Falkena: <http://www.mathworks.com/matlabcentral/fileexchange/28518>

These functions are copyrighted by their respective authors and licensed under the BSD license. For convenience, they are included in the GEMV² package in the `+externalCode` directory, along with their respective licenses. Some functions were modified to suit the purpose of GEMV². Google Earth Toolbox and OpenStreetMap Functions are elaborate toolboxes with many functionalities not required by GEMV². Therefore, the minimum set of required functions from these toolboxes is included in GEMV² package. The original versions of functions can be downloaded from the MATLAB Central File Exchange as indicated above.



2.5 What is in the package?

The complete GEMV² .zip package contains the following:

1. GEMV² code.
2. GEMV² license.
3. This user manual.
4. Dataset collected via aerial photography, containing buildings and vehicles in the city of Porto, Portugal, obtained through the DRIVE-IN project and described in Ferreira et al. [4]. This dataset is highly precise and can be used as a starting point for the simulations.
5. Since Porto dataset contains only a single time-step, in addition to it, packaged with GEMV² you will find building and foliage outlines extracted from OpenStreetMap [5], and SUMO mobility for the following cities:
 - Part of downtown Cologne, Germany. For vehicular mobility in Cologne, the package contains a preprocessed SUMO [6] file generated using TAPAS Cologne scenario. Running GEMV² with this dataset results in 100 simulated seconds.
 - Dataset with 20 simulated seconds in Newcastle, UK.
 - Dataset with 50 simulated seconds in Bologna, Italy. This dataset contains both V2V and V2I communication.

3 Building and Foliage Outlines

3.1 Using Outlines of Buildings and Foliage from OpenStreetMap

The outlines of buildings and foliage are available from OpenStreetMap [5], which is the most comprehensive free geographical database. That said, the actual representation of buildings and (particularly) foliage in the OpenStreetMap can be limited, with many objects missing. You need to be judicious where you perform simulations, since a database missing a lot of buildings and foliage will likely result in unrealistic simulation output.

Buildings and foliage can be imported into MATLAB using the OpenStreetMap Functions package available at <http://www.mathworks.com/matlabcentral/fileexchange/35819>. A version of the package has been included with the model. It was modified to output a three-column array [objectID | Latitude | Longitude], each row representing one point of an object.

To use buildings and foliage from OpenStreetMap other than those already included in GEMV² package, simply go to <http://www.openstreetmap.org> and

export the objects for a desired location. Then, point the variable `staticFile` in `simSettings.m` to the downloaded `.osm` file. This will invoke the OpenStreetMap parser and save the processed data into an array, so that the conversion is performed only once.

If you plan to run the simulation over a large area (e.g., an entire city), follow these steps:

1. Download `.osm` or `.pbk` file containing desired simulation area from sources indicated on the OpenStreetMap website: http://wiki.openstreetmap.org/wiki/Downloading_data
2. Get the `.osm` file for the specified region (defined with southwestern and northeastern bounding points) using `osmconvert` (http://wiki.openstreetmap.org/wiki/Osmconvert#Clipping_based_on_Longitude_and_Latitude):

```
osmconvert bigArea.pbf -b=-75.8,45.1,-75.7,45.2 > cityArea.osm
```
3. To further reduce the `osm` file, you can filter out unnecessary objects using `osmfilter` (<http://wiki.openstreetmap.org/wiki/Osmfilter>):

```
osmfilter cityArea.osm --keep="building" --keep="shop"
--keep="amenity" --keep="landuse" --keep="forest"
--keep="natural" --keep="wood" --keep="tree"
--keep="tree_row" -o=cityAreaPoly.osm
```

In practice, the input `.osm` file containing buildings and foliage can be up to approximately 50 MB in size. This limitation is mainly due to `xml2struct` function (www.mathworks.com/matlabcentral/fileexchange/28518), which requires approximately 5-10 times the size of `.osm` file in memory when converting the `.osm` file to MATLAB structure).

3.2 Converting Shapefiles to OpenStreetMap

If you have buildings and foliage in shapefile format, to use them with GEMV², one option is to first convert the data to OpenStreetMap format. This can be done using the `shp-to-osm.jar` utility available at <http://wiki.openstreetmap.org/wiki/Shp-to-osm.jar> (related GitHub project is located at: <https://github.com/iandees/shp-to-osm/downloads>). The conversion has been successfully tested with version `shp-to-osm-0.8.6-jar-with-dependencies.jar` (available on GitHub). Example of usage:

```
java -cp shp-to-osm-0.8.6-jar-with-dependencies.jar
com.yellowbkpk.geo.shp.Main --shapefile inputPolygons.shp
--rulesfile RULES.TXT --osmfile outputOSM.osm
--outputFormat osmc --maxnodes 100000000
```

Once the OSM files have been generated, they can be converted to the simulator-readable array as explained in Section 3.1.

4 Vehicular Mobility and Outlines: SUMO Floating Car Data

Vehicular mobility generated by SUMO can be used as input for GEMV². The function `parseSUMOMobility.m` converts SUMO XML output to a five-column array [ID | Latitude | Longitude | vehicleType | bearing], which is then converted to the three-column array [objectID | Latitude | Longitude] using the function `generateVehiclePolygons.m`.

First, export the vehicular mobility from SUMO by using `-fcd-output` option and outputting latitudes and longitudes. For example, in Windows:

```
sumo.exe -c sumoCfgFile.cfg --fcd-output SUMOMobility.xml
--fcd-output.geo true
```

Next, use `SUMOMobility.xml` as input to GEMV². In terms of the size of the input SUMO mobility file, same considerations apply as in Section 3.1.

5 Referencing GEMV²

If you find GEMV² useful in your work, please cite the following reference [1]:

Mate Boban, João Barros, and Ozan K. Tonguz: “Geometry-Based Vehicle-to-Vehicle Channel Modeling for Large-Scale Simulation,” IEEE Transactions on Vehicular Technology, Vol. 63, No. 9, November 2014, doi:10.1109/TVT.2014.2317803 [PDF][BIB]

If you primarily use “vehicles as obstacles” model implemented as part of GEMV², please cite the following reference [7]:

Mate Boban, Tiago T. V. Vinhoza, Michel Ferreira, João Barros, and Ozan K. Tonguz, “Impact of Vehicles as Obstacles in Vehicular Ad Hoc Networks,” IEEE Journal on Selected Areas in Communications, vol. 29, no. 1, pp. 15–28, January 2011, doi:10.1109/JSAC.2011.110103 [PDF][BIB]

Similarly, if you are primarily using the V2I propagation modeling, please cite the following reference [8]:

Bengi Aygun, Mate Boban, Joao P. Vilela, and Alexander M. Wyglinski: “Geometry-Based Propagation Modeling and Simulation of Vehicle-to-Infrastructure Links”,

under submission

Acknowledgements

This work was funded in part by the Portuguese Foundation for Science and Technology under grants SFRH/BD/33771/2009 and CMUPT/NGN/0052/2008.

References

- [1] M. Boban, J. Barros, and O. Tonguz, “Geometry-based vehicle-to-vehicle channel modeling for large-scale simulation,” *IEEE Transactions on Vehicular Technology*, vol. PP, no. 99, pp. 1–1, 2014.
- [2] D. G. Bell, F. Kuehnel, C. Maxwell, R. Kim, K. Kasraie, T. Gaskins, P. Hogan, and J. Coughlan, “NASA World Wind: Opensource GIS for mission operations,” in *IEEE Aerospace Conference*, 2007, pp. 1–9.
- [3] M. Boban, “Demo: Visualization of vehicular communication: Insights into power, effective range, clustering, and neighborhood size,” in *IEEE Vehicular Networking Conference (VNC 2014)*, December 2014, pp. 205–206.
- [4] M. Ferreira, H. Conceição, R. Fernandes, and O. K. Tonguz, “Stereoscopic aerial photography: an alternative to model-based urban mobility approaches,” in *Proceedings of the Sixth ACM International Workshop on Vehicular Inter-NETworking (VANET 2009)*. ACM New York, NY, USA, 2009.
- [5] M. Haklay and P. Weber, “OpenStreetMap: User-generated street maps,” *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, 2008.
- [6] D. Krajzewicz, G. Hertkorn, C. Rössel, and P. Wagner, “SUMO (simulation of urban mobility),” in *Proc. of the 4th Middle East Symposium on Simulation and Modelling*, 2002, pp. 183–187.
- [7] M. Boban, T. T. V. Vinhoza, M. Ferreira, J. Barros, and O. K. Tonguz, “Impact of vehicles as obstacles in vehicular ad hoc networks,” *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 1, pp. 15–28, January 2011.
- [8] B. Aygun, M. Boban, J. Vilela, and A. Wyglinski, “Geometry-based propagation modeling and simulation of vehicle-to-infrastructure links,” in *paper under review*, 2015.