Reducing the implementation uncertainty using an advanced robotic simulator

Gabriel Burtin[1], Florent Malartre[1], Roland Chapuis[2]
[1]4D-Virtualiz, 10 allée Evariste Galois, France gabriel.burtin@4d-virtualiz.com
florent.malartre@4d-virtualiz.com
[2]Institut Blaise Pascal, 4 impasse Blaise Pascal, France roland.chapuis@univ-bpclermont.fr

Keywords: robot, simulation, real-time simulation, control, multi sensor, data fusion, time consistency

Abstract: This paper addresses the problem of crossing the gap between pure scientist theory and real robotic applications by the use of a robotic simulator. A step by step process can ease this transition. Those steps provide the ability to graduate the realism of the simulation. The user can begin with a simple and not realistic simulation, and evolve slowly toward more complex and realistic simulation. To perform this gradation of the simulation, the sensors and actuators must be designed to be able to modulate their realism level. To complete the sensors models, the simulator provides virtual communication protocols, identical to the real ones: the *plug-and-play* feature. Those interface simulation helps the final transition when the researcher un-plugs the simulator from the intelligent system and plugs it to the robotic system in the real world. A simulator may not be able to achieve hard real-time and so correctly interface with real-time intelligent system. Therefore, to ensure a realistic simulation, a simulator must include a time consistency mechanism. In this paper the realism of the simulated sensors has been compared to a real dataset, created on PAVIN, an experimental site. Another test, performed only with the simulator, gave us the consequences of the time-consistency mechanism implementation, allowing a more realistic simulation.

1 INTRODUCTION

Every robotic application starts with an idea, this idea has to be tested, developed and validated. There is a process (Figure *1*) between the original idea and the real application. This process goes through virtual and field validation.

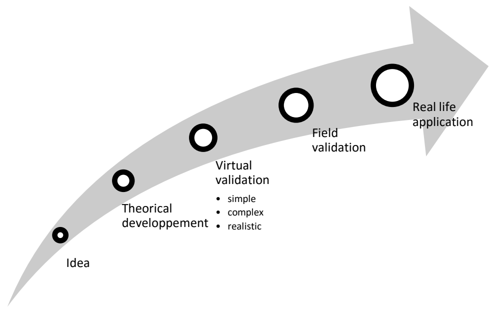


Figure 1: Evolution of an idea toward the application.

Applications fields are numerous and various: agriculture, flight systems, freight, intervention in hazardous zones (civilian or military), etc. These various fields imply a lot of tests in various environments with a great diversity of robots.

Laboratories and companies massively use real robot platforms to test and benchmark every interesting combinations of sensors / algorithms / environments.

However, this methodology has its limits. It's necessary to own the sensors and the robots (buying or loaning the materials), or to be able to access it if the materials are shared with other persons and then calibrate and configure everything (time consuming step). Finally, after all these steps, if the weather conditions are propitious and the algorithm is bug-free, you can proceed and realize the final test.

2 CONTEXT

During their thesis, two PhD students (Malartre, 2011) (Delmas, 2011) met these issues: the urge to test many use cases, being dependant of the robot's availabilities, being weather dependant, not being able to afford every sensors and easily connect to them. They decided to use their robotic experience and develop their own real-time robot simulator: 4DV-Simulator.

One of the goals of this simulator is to reduce the gap between simulation and reality by providing an accurate simulation of the sensors and actuators.

This gap is one of the most important obstacles due to the differences between simulated and real world. Every failure during field testing is delaying the project. The researcher could lose time, changing the framework operating system, code, and the interfaces of the sensors. Therefore, if the application is enough mature before testing it on the field, the project ends faster.

The solution to reduce the overall gap is to split the development with smaller gaps. To do so, every gap need specific elements to validate specific constraints. Several robotic simulator exists, here is a non-exhaustive list: MATLAB®-Simulink, MORSE, V-REP, Gazebo, etc. They are specific to a use case. Simulink runs simple and small simulations with common shapes. V-REP is dual licenced (GNU GPL/Proprietary) and propose several physics engines and a large set of API. MORSE and Gazebo are open-source simulators, able to simulate several robots but interface mainly through middleware (ROS, YARP etc.) and give an average sensor modelisation.

The advantage of 4DV-Sim is the ability to carry a project from the beginning of the conception to the transition toward the field tests (Figure *2*). Using gradated complexity and the *plug-and-play* feature implemented in this simulator, the transition between each level of complexity won't require any time consuming efforts.

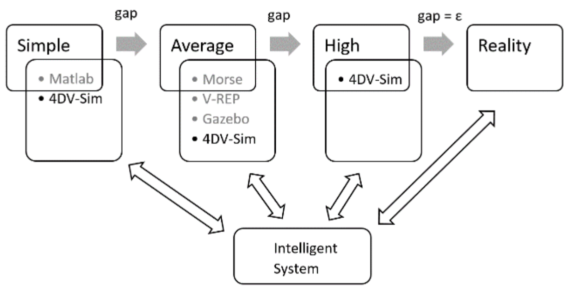


Figure 2: Existing simulators for different simulations realisms.

3 GRADUATED REALISM

3.1 Sensors models

Any intelligent system must be fed with sensor's data to work properly. These data may come from either a real or a simulated sensor.

For example, the "beam" sensor, provided in the robotic toolbox in (Corke, 1996) is related to a real sensor. But this simulated sensor doesn't have a realistic behavior: there is no noise, it doesn't provide false positive etc. This basic simulated sensor is enough for basic testing such as a simple Kalman filter and SLAM navigation in a MATLAB® environment.

If the intelligent system is likely to be used in complex environment, it is important to split the development process to lower the complexity of every step. Each step is assigned a complexity level.

To fit with the complexity level, the sensor realism has to evolve accordingly. This implies a sensor with gradated complexity or gradated realism.

In our case, we will focus on the GPS sensor.

A very rough model of a GPS would be to directly use the ground truth from the simulator (X, Y, Z), then apply the inverse of the appropriate projection and obtain perfect GPS localization using WGS84 coordinate system (http://earth-info.nga.mil/GandG/publications/tr8350.2/tr8350_2.html).

A slightly more realistic model would be to use the manufacturer data. The company provide the technical data and the accuracy of the sensor: mean error, noise, drift etc. With these information, it's possible to alter the GPS data by adding a noise and a drift to the "perfect" data given by the previous procedure.

This simulation approach, is not realistic enough: to provide realistic simulation of any sensors, it is necessary to come to the basic of the sensor, the way it works physically in the real world.

The GPS system has satellites orbiting above the earth, sending signals. Then, a GPS receiver needs at least 4 satellites signals to obtain an accurate localization. Therefore, the next level of model is the use of simulated satellites, sending simulated signals (the signal goes through the different atmospherics layers at different speeds) to the simulated GPS sensor to compute its position. This way, if the GPS discovers a new satellite or lose a signal, the position's accuracy will vary.

Now that we introduced satellites, we can assign random position to the satellites or we can position the satellites precisely. Using the year, month, day and time of the simulation, a specific library provides the exact position of the real satellites (Tolman, 2004). By setting the satellites at the real position, the simulated constellation is exactly the same as the real one. Thus the satellites will appear and disappear at the same moment.

The physical obstacles in the environment can block or reflect the GPS signal which will interfere with the accuracy of the localization. To perform the most realistic GPS model, even the signal loss and multi-trajectories are implemented. When the robot with a GPS travels between buildings, the GPS signal is more likely to be reflected by the wall, affecting the precision of the localisation (Dana, 1997) (Laneurit, 2006). This had to be implemented to provide the most realistic GPS model.

Therefore, we can provide a gradation of the GPS realism (Table 1). A realism level of 0 is a non-realistic GPS with perfect localization. A realism level of 4 is the most realistic GPS, including every perturbation cited above.

Table 1: Gradation of the GPS complexity.

Realism level	Simple noise	Drift	Constellation	Ephemerid	Signal loss (indoor)	Multi-trajectory
0						
1	✓					
2	✓	✓				
3	✓	✓	✓	✓		
4	✓	✓	✓	✓	✓	✓

Any sensor is likely to be created with a graduated complexity. For example, camera has intrinsic parameters. They can be simple using the pin-hole model, with radial and tangential distortions models (Claus, Fitzgibbon, 2005), or even more complex models: affine, omnidirectional (Banerjee, 2001) (Scaramuzza, 2014).

The lidar sensor is also important to be realistic: many state of the art robotic applications use lidar based SLAM. A basic model would be to draw lines all around the sensor to compute the distances at a given moment. But this is not realistic when the sensor is mounted on a mobile robot: most of lidars have a rotating mirror to provide the scanning mechanism. The angular speed is finite and the sensor need a specific amount of time to scan from the first acquisition to the last. The lidar sensor doesn’t have the same position at the beginning of the scan and at the end. A rolling-shutter-like phenomenon appears and deserves to be included in the sensor’s model. Those are examples of the gradated sensors implemented in this robotic simulator.

This ability to increase the complexity is an advantage to accelerate the development of robotic applications: the researcher can test its intelligent system without changing the framework.

After several iterations, when the intelligent system works with the highest simulation realism, the gap between simulation and reality is as small as possible.

3.2 Plug & Play feature

The second barrier to make the transition between the simulated world and the real world is how to interface the data with the targeted hardware.

By using any framework, the researcher needs to elaborate the appropriate interface: Simulink, ROS, YARP etc. When he does the transition between the simulation and the real robot, he has to re-develop the drivers/interfaces to connect to the real sensors or actuators on the existing platform.

4DV-Simulator also provides a transparent interface to ease the transition toward the real world.

The objective is to connect the intelligent system transparently between the simulated sensors and the real sensors by emulating the interface.

To perform this action, it is necessary to reproduce both components of any interface: the medium and the protocol.

The medium is the physical way over which the information is transported. For example, some of the existing media are: Ethernet, Wi-Fi, Serial port, CAN, FireWire, USB, RF. Every support has specific parameters to be configured to work properly: IP address; port, parity bit, end bit, baud rate, frequency, channel, etc.

The protocol defines the form of the data that is going to be exchanged on the medium. For example, Figure 3 is a part of the byte sequences for the Xsens® sensor protocol.

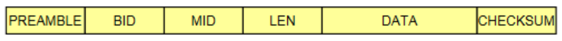


Figure 3: MTi-G sensor from Xsens®.

This Xsens® protocol is constituted of a header (preamble), a bus ID, a message ID, the length of the data, the data and finally a checksum.

A protocol is more likely to be specific for a given sensor.

The simulator gives the ability for the user to directly use the real protocol from the very beginning of the project. The simulated sensor will send the data through the selected virtual interface (aka serial RS232 link with the Xsens® sensor: Figure 4) and send it outside the simulation platform to the intelligent system.

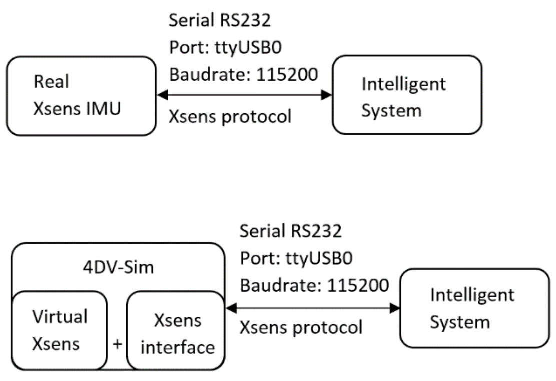


Figure 4: Xsens IMU interface comparison between reality and simulation.

If the intelligent system has the ability to connect itself to the simulated sensor, it will be able to connect to the real system without any extra modifications (this feature can be seen on a video at: https://www.youtube.com/watch?v=zCePWUhM62U).

This interface emulation guarantees that the work won't have to be done several times to develop the robotic application interfaces. The development can therefore focus on the intelligent system. The overall development is accelerated and eased, the gap between simulation and reality is reduced to a minimum.

4 TIME CONSISTENCY

Now that the overall sensor model has been tackled (sensor + interface), it is necessary to pay attention to another simulation related issue: time.

The evolution of time, in real life, is linear, fluid (putting aside the relativity). Inside a simulator, the time evolves at a certain time step. This time may be constant or dynamic. Therefore, we can encounter two types of simulations: non real-time and real-time.

The first one is proposed by tools such as Catia®, Abaqus®, SolidWorks® etc. The computer has "infinite" time to compute the transformation between two time steps. These simulators can achieve high accuracy scenario using solvers with high numbers of iterations. However, robotics applications require the interaction of software modules with different frequencies, eventually synchronised. These software modules are the sensors, actuators, intelligent systems, decision making software etc. They interact with the environnement. With this simulator all these modules are modelled: sensors, actuators and physics reactions as well. To reach an accurate model, the simulator has to run in real-time. This means that when 1 second is elapsed in the reality, the time in the simulator must be elapsed of 1 second.

This constraint requires very fast computation and heavy multi-threading to be able to perform the smallest time-step and keep up with the real-time. If the time-step is too big, the simulation is not realistic (undetected collisions, etc.) but if the time step is too small, and the system can't be real-time, then the simulator will accumulate latency with the real world.

For example, a simulator with a 50% real-time performance will need 2 milliseconds to compute a 1 millisecond time-step. This difference introduces a lag between the simulator and the reality.

This lag may cause issues for control loop process when a real-time intelligent system is connected to the simulator.

For example, if at time 't', the simulator sends a data to the intelligent system (we assume in this example that the data transportation is instantaneous). At his exact moment the simulator efficiency drops to 50% of real-time. The intelligent system spends a (real-time) duration 'Δ' to compute the data and to generate a command, sent to the simulator. Unfortunately, because of the 50% efficiency of the simulator, the timeline is not the same between the real world and the virtual world. The data is sent back at time = t + Δ in the real world, but at t + Δ/2 in the virtual world (Figure 5). If the simulator doesn't possess a time consistency mechanism, the information will be integrated in the simulator too early compared to when it should.

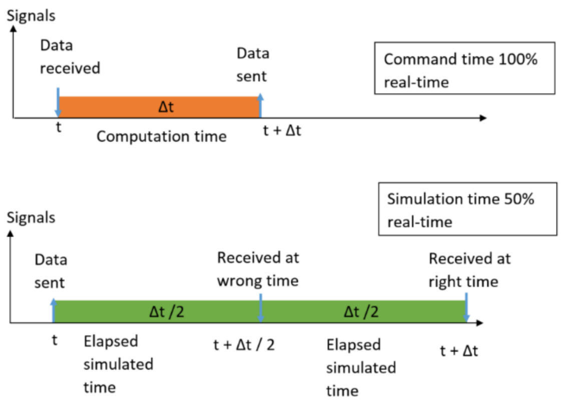
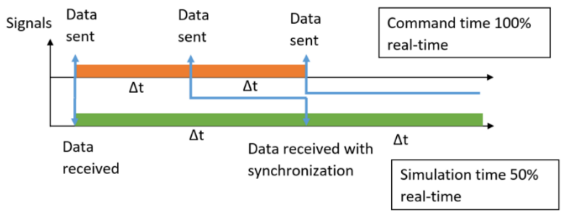


Figure 5: Evolution of the time inside and outside a simulator.

This error can be avoided using a time consistency mechanism (Delmas, Malartre, 2015). Using the time-stamp of the generated data, the simulator can re-synchronise (Figure 6) the received data and use it at the exact moment it was meant to be received and used.


Signals
Data sent
Data sent
Data sent
Command time 100% real-time
Δt
Δt
Δt
Δt
Data received
Data received with synchronization
Simulation time 50% real-time
Figure 6: Synchronization of the data.

This essential mechanism has patented and has been implemented in 4DV-Simulator. Experimentation and results are presented in section 5.2.

5 EXPERIMENTATION AND RESULTS

We ran two experiments to evaluate the efficiency of sensor models and the time consistency mechanism. We will compare the differences between real and virtual, then we will observe the impact of the time consistency mechanism.

We compared a real dataset with a dataset given by simulated sensors in a simulated environment in section 5.1. We will describe the experimentation process in sub-section 5.1.1. In the following sub-section (5.1.2), we will show the results and discuss them.

In section 5.2, we will observe the effects of a given control law, controlling a simulated vehicle, with and with-out time consistency. In sub-section 5.2.1 we give the parameters used to test the time consistency mechanism, and the results will be discussed in sub-section 5.2.2.

5.1 Comparison between real and simulated data

5.1.1 Experimentation process

To test the realism of the sensors, we used a dataset provided by Blaise Pascal institute (Korrapati, Courbon, Alizon, Marmoiton, 2013). This dataset was produced with the experimental site PAVIN located on the campus “les Cezeaux” in Clermont-Ferrand. The site is a 50mx100m urban area with road junctions, floor markings, traffic lights and scaled buildings.

The experimental dataset contains time-stamped data from several sensors embedded on a robotic platform: camera, RTK GPS (ground truth), low cost GPS, IMU, odometry and lidar.

The ground truth allowed us to recreate the exact trajectory the vehicle travelled on site.

The sensors configuration has been reproduced in the simulator with virtual sensors.

A 3D model of this experimental site has been used into our simulator and virtual vehicle has been equipped with the same sensors (camera, GPSs and lidar). It followed the trajectory given by the real vehicle. The objective is a qualitative comparison between the real data from the data set and the data produced by the virtual sensors.

5.1.2 Results

The Effibox software have been used to record and display the data from the real and virtual sensors (as provided by the dataset website).

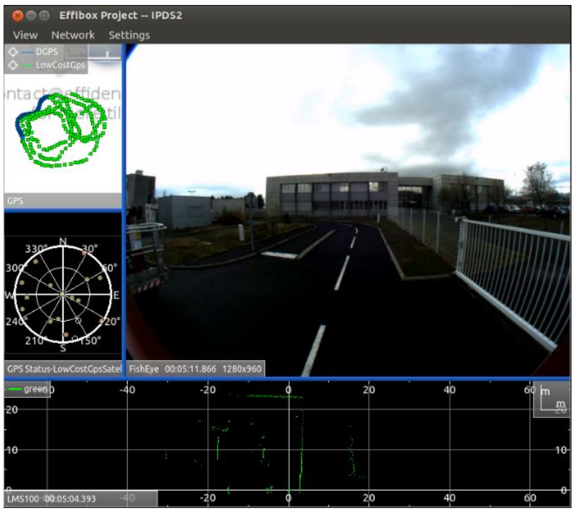
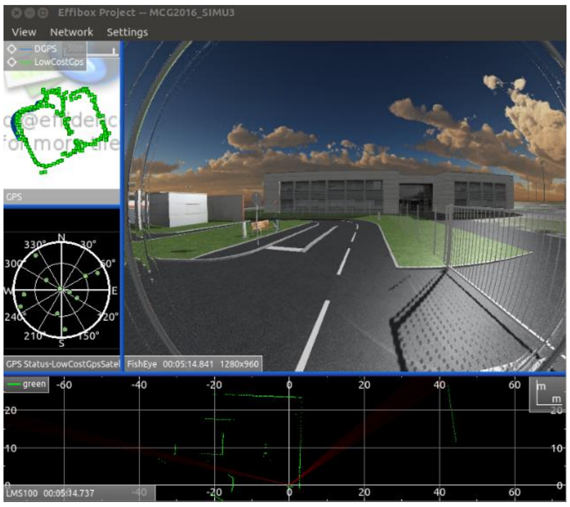


Figure 7: Real configuration

We can observe the data provided by a fisheye camera, a low cost gps, a RTK gps and a lidar (Figure 7 and Figure 8). The real gps receiver uses the GPS and GLONASS satellites constellation (Figure 7) while the virtual uses the GPS only (Figure 8). Therefore, a few satellites are shown on the real and not shown on the virtual constellation.

The simulator can provide a realistic simulation of the pose of the sun at a certain hour and day of the year, giving realistic shadows with buildings, fences (Figure 8). Unfortunately, the clouds can’t be detemined by these parameters and they are not included in the simulation. Therefore, the luminosity of the real exprimentation is lower than the simulated, but we can clearly observe the similarity between the camera images and the fisheye distortion.


Effibox Project -- MCG2016_SIMU3
View
Network
Settings
DGPS
LowCostGps
GPS
GPS Status-LowCostGpsSatel
FishEye 00:05:14.841 1280x960
LMS100 00:05:14.737
Figure 8: Virtual configuration

Another effect, unpredictable, is the position and composition of the clouds. They may be able to alter the velocity of the signal (troposphere errors).

5.2 Time consistency mechanism

5.2.1 Experimentation process

To perform a comparison between a real time simulator with and without time consistency mechanism, we created a separate and simple program sending steering and speed commands to a virtual robot. In the first case, the program will use the time consistency mechanism, in the second case the program will send raw command values without taking into account the time consistency.
The simulator was artificially slowed down to run at 50% real-time speed.

5.2.2 Time consistency results

The following graph represents the commands received by the robot inside the simulator. The linear velocity command is concerned by the time consistency phenomenon. The "dash-dot" curves are the commands without time consistency mechanism, the "dash-dash" plot are the commands with the time consistency mechanism (Figure 9).

The computation time assigned to the command algorithm is 15ms. We can observe a significant difference between the data used with and without the time consistency mechanism at the beginning of the simulation and this differences amplifies over time due to the open loop configuration of the command. This difference introduces an error in the control law and obviously the final pose of the robot was totally different (Figure 10). The final error is 17cm after a 30m trajectory. In case of using a closed loop control law, oscillations should appear and amplifies the error.

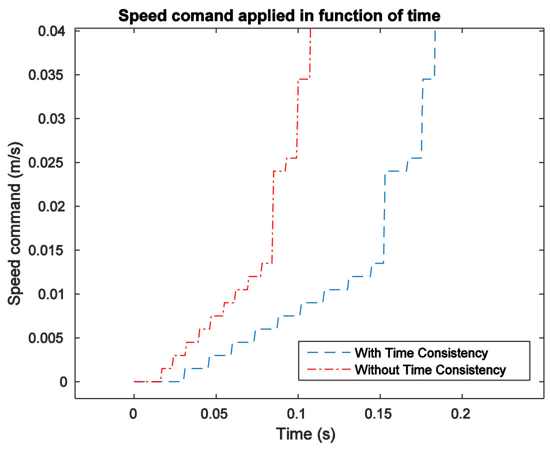


Figure 9: Comparison of the applied velocity commands at the beginning.

The time consistency mechanism introduces the command at the correct moment : the simulator is slowed and therefore, the command is used later .

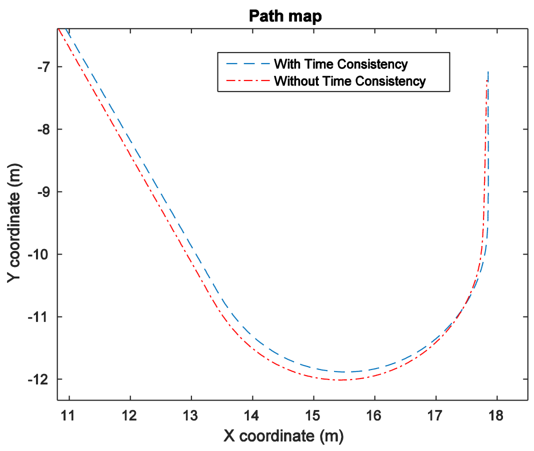


Figure 10: Zoom on the end of the vehicles trajectories.

6 CONCLUSIONS

In this paper we presented a fastening approach for robotic developments. The research and development is segmented with different levels of complexity to lower the gap and clear every technological lock, one by one. To segment this development, we used only one tool, flexible and accurate, to model every level of the required complexity. These levels are obtained by proposing a set of sensors and actuators with graduated complexity. Along with the *plug-and-play* feature, this allows the researchers to easily test and validate different aspects of a designed intelligent system.

All these features are exclusively implemented in the 4DV Simulator.

As prospect, we would like to take advantage of this realistic modelling of the sensors and the environment to increase the number and the quality of the graduated sensors. We hope to be able to define appropriate metrics to compare and match real and virtual datas. Along with a high fidelity replicated environement, it could be used to run real-time comparison between real and virtual world, improving robot navigation.

ACKNOWLEDGEMENTS

The authors would like to thank the Institut Pascal for providing the sensors datasets created in PAVIN.

REFERENCES

Malartre, F., 2011. Thèse soutenue le 16 juin 2011, "Perception intelligente pour la navigation rapide de robots mobiles en environnement naturel", Université Blaise Pascal II, Aubière, France.

Delmas, P., 2011. thèse soutenue le 24 février 2011, "Génération active des trajectoires d'un véhicule agricole dans son environnement", Université Blaise Pascal II, Aubière, France.

Corke, P., 1996. A robotics toolbox for MATLAB. IEEE Robotics & Automation Magazine, 1996, vol. 3, no 1, p. 24-32.

Tolman, B., Harris, B., 2004. "The GPS Toolkit." Linux Journal. September 2004, p. 72.

Dana, P.H., 1997. "Global positioning system (gps) time dissemination for real-time applications", The International journal of time critical computing systems, 12(1): 9-40, 1997

Laneurit, J., 2006. Perception multisensorielle pour la localisation d'un robot mobile en environnement extérieur, application aux véhicules routiers (Doctoral dissertation, Université Blaise Pascal-Clermont-Ferrand II).

Claus, D., Fitzgibbon, A., W., 2005. A rational function lens distortion model for general cameras. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* (Vol. 1, pp. 213-219). IEEE.

Scaramuzza, D., 2014. Omnidirectional camera. In *Computer Vision* (pp. 552-560). Springer US.

Banerjee, S., 2001. Camera Models and Affine Multiple Views Geometry. New Delhi: IIT Delhi, 2.

Delmas, P., Malartre, F., 2015. "Système et procédé de test d'un organe machine ou robot", Brevet Fr 15/01121, dépôt 01/06/2015 par 4D-Virtualiz.

Korrapati, H., Courbon, J., Alizon, S., Marmoiton, F., 2013. "The Institut Pascal Data Sets": un jeu de données en extérieur, multicapteurs et datées avec réalité terrain, données d'étalonnage et outils logiciels. In ORASIS, June 2013, Cluny, France.