System Dynamics Inspired Sensor Modeling and Simulation

Sören Schweigert
OFFIS e.V.
Oldenburg, Germany
Soeren.Schweigert@offis.de

***Abstract*—Applications within the automotive and robotics domain highly depend on the correct sensor perception. In order to validate the partial safety-critical requirements for sensor processing, simulation tools are prevalent. The comparison between simulated and perceived environments allows conclusions about the quality of the sensor processing. To obtain accurate quality estimations, the simulation has to provide realistic sensor measurements. This is important in order to facilitate a subsequent integration with the physical sensors. In this work a system dynamics inspired modeling and simulation approach is presented, that allows describing both the sensors, as well as the so far often neglected environmental conditions and sensor interferences. In addition the model is capable to be transferred into a context specific shader program at simulation runtime to enable fast and efficient computation on the used graphics hardware.**

Keywords-sensor simulation*; *system dynamics*; *ground truth generation*; *error models

I. Introduction

Autonomous guided vehicles (AGV) are increasingly used in logistics and factory automation. Currently these systems operate either in closed environments like factory halls with limited access or have to operate at very low speed to ensure the safety of other participants, e.g., humans or other (human guided) vehicles. To increase the speed and thus the effectiveness, while maintaining a high safety level, the AGV handles many sensor data in order to construct an image of the environment. Some commonly used sensors for this task include both; optical sensors such as cameras, and rangefinders such as laser scanners, ultrasound or Photonic Mixing Devices (PMD cameras).
Since the sensor perception is essential for the safety aspects, it has to be comprehensively tested under various conditions. As testing in the real world is time-consuming, expensive, and especially in the first development phase risky, sensor simulations can be used for early tests. In addition simulation provides the possibility to compare the captured environmental image against the simulated environment and thus allows evaluating the sensor processing. However to obtain a realistic validation for the processing quality, the simulated sensors have to provide realistic sensor measurements, taking into account different error models and environmental influences.

The rest of the paper is structured as follows. Section II discusses requirements for sensor simulations in general and in terms of quality. These requirements are compared to currently available simulations. In Section III a modeling approach is introduced, that allows to model environmental influences on sensor measurements to fulfill the requirements discussed in Section II.

II. Requirements

To use simulation for AGV development, certain requirements must be met. Some of these requirements concern integration issues; others are related to simulation quality, yet others are useful for testing purposes.

A. Integration requirements

Simulations are used to develop AGV in a safe and convenient environment. This does speed up the development process since a lot of technical issues do not need to be considered in the early stages. However, when porting the developed algorithm from simulation to real hardware integration issues on how to **access resources** like laser scanners or cameras arise. To address this problem, most simulation environments skip along with a sensor/actor middleware that harmonizes the access to simulated and real hardware. One of the most famous representatives of this technique is the Player/Stage/Gazzebo project, started in 2000 [1]. The Robot Operating System (ROS) [2] and the Mobile Robotic Programming Toolkit (MRPT) [3], mainly use the same approach. The major drawback of this technique is that the developer is **forced to use a given architecture**.
Another approach is to recreate the interfaces of the real sensor within the simulation, including the real protocols. The described simulation uses a combination of both approaches. If possible, we recreate the interface provided by the real sensor. If the recreation is not possible, for example due to hardware limitations, a self-designed data structure is used, that allows easy transformation to other formats.

B. Testing requirements

Using simulation allows to use a user defined test environment, with all states formally described. This information should be used to provide **symbolic information** about the objects within a certain area or an **accurate occupancy grid** (containing only free or occupied cells) within the

 ISBN: 978-1-61208-234-9

environment. This information can be collected manually, as it is done in the Tunnel Simulator [4] or extracted from the simulation.
To test the quality of a certain sensor processing algorithm additional information about the sensor **ground truth** should be provided. For example the disparity map for a stereoscopic camera or a symbolic representation of all objects within a camera image.

A useful but yet not essential requirement is **real time performance**. If the simulation can provide real time sensor measurements, the performance of complex sensor processing pipelines can be measured. Faster than real-time may allow the usage of optimization or learning algorithms on both sides the sensor processing or the simulation itself. For example to learn the boundaries of the processing algorithms.
Some simulation platforms like Player/Stage/Gazzebo or the simulation component of the MRPT are using a simplification of the sensor data generation process, to gain more than real-time performance. Rossmann et al. [5] follows an interesting approach, to use VR-Hardware for efficient and accurate laser scanners. Others like the RoboCup-Rescue League simulator USARSim [6] are designed to run in real-time. Yet others like VANE [7] or the Tunnel Simulator produce highly accurate sensor measurements but cannot simulate in real-time.

C. Quality requirements

The most important factor for testing is the quality or accuracy of the provided sensor measurements. Since simulation tends to generate ideal measurements, real sensors do not. The following subsection will discuss some of the influences according to sensor measurement generation accuracy.

When talking about quality of virtual or simulated sensors, it has another meaning than the sensor processing community. In case of simulation, sensor quality means the availability to masquerade the existence of the simulation, by providing realistic sensor measurements, like described in Siegel et al. [8]. Within the real world there are several factors, having an impact on the quality of sensor measurements. These Factors need to be considered, when creating a sensor simulation:

1) stochastical noise
2) systematical noise
3) context sensitive noise
4) environmental influence
5) influence of other (active) sensors

Whereas the first two factors are commonly known and may be caused by thermal noise, in case of **stochastical noise**, or due to small errors in sensor-production and ageing effects, in case of **systematical noise**. (A more detailed description for these errors can be found in [9].)
The other factors are quite complex and often ignored during sensor simulation.

Context sensitive noise refers to the current situation within the simulation. Meaning, these errors result through interaction with other objects inside the simulation environment. In most cases they are reasoned by the physical principle of taking measurements. One example is an ultrasonic sensor mounted on a vehicle at low height. Even if the expected range is $6m$, the device could measure the ground at a distance of $1m$. As result the effective range for this device would be less than one meter. Similar examples for laser scanners can be found in Goodin et al. [10] where the beam divergence is taken into account. To observe these effects inside the simulation, the physical principle implemented in the real sensor needs to be matched as close as possible.

Each off these effects considers only one instance of the simulated sensor. However **environmental factors** like rain, fog and temperature do not affect only one sensor. Thus if the sensor processing detects that the range of a laser scanner is reduced because of fog, a camera near to this scanner should also indicate the existence of fog, to support reasoning algorithms.

The last effect, **influence by other (active) sensors**, can easily be observed when using the Microsoft Kinect sensor, emitting an infrared dot pattern [11]. This pattern can influence normal Webcams and thus have an important effect for computer vision algorithms that may be part of the processing pipeline that is under test.

III. Modeling approach

This section provides a general overview of the proposed modelling approach and how it is intent to solve the above discussed requirements, especially the quality requirements. A distinction is made between simulation and virtual sensors. While the simulation is used to generate an accurate representation of the environment, the sensors perceive the provided data to calculate their return values. It follows that the measured values always refer to an object in the simulated environment. In case of optical sensors it can be broken down into specific points on the surface of a three dimensional object. A prominent example is the light calculations in modern computer games which simulate the interaction between a light source and a surface. The usual approach is to manually define a computation rule for each surface. Since 3D games often consist of a variety of static and dynamic lights the defined computation rule needs to consider all kinds of light sources. Modern 3D engines like OGRE [12] use shader generators to automatically create such computation rules.

This technique should be applied to other sensors than cameras and other emitters as light sources. Since each

 ISBN: 978-1-61208-234-9

sensor needs a different input for a reasonable calculation of sensor measurements, a modeling methodology is introduced inspired by system dynamics. [13]
The model consists of three sub-models which are converted automatically into a computation model that is used to generate shader code during runtime of the simulation. Thereby the current context of the sensor and its environment is taken into account. The tree sub-models are:

1) Emitter Model: Defines the factor of influence (FOI) that is required to perform the calculation for at least one class of sensors.
2) Sensor Model: Defines how the specified FOI needs to be combined for one sensor class, thus defines the sensor itself.
3) Sensor Instance Model: This model allows a differentiation between two sensors of the same class. It mainly defines a post processing step, and allows to define error models for individual sensors.

A. Emitter Model

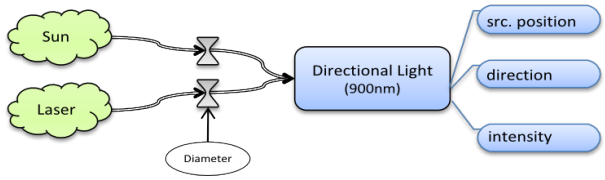

Figure 1. Example Emitter model containing two emitter classes, that do both emit directed light with a wavelength of $900nm$.

The emitter model defines an FOI that is required by at least one class of sensors as input. It is associated with all object classes which have an influence on this factor. As can be seen in Figure 1. Each factor may consist of several attributes. In the example, the directed light is determined by its source position, direction and intensity. An emitter is a class of objects that have an impact on the FOI in a certain area. They serve as global input variables in the model and thus represent the parameter of the simulation. In most cases they are associated with a 3D object within the simulation containing some basic properties like position, and orientation. Their emitted values can be described by either a constant value or a temporally resolved function. In Figure 1, there are two FOI, the sun that represents a temporarily resolved function, which provides intensity over one day, and the Laser range finder, that emits a light beam of constant intensity. The auxiliary node describes how the emitted value can be transformed into the FOI. This is done by functions for each attribute of the FOI.

B. Sensor Model

The sensor model describes how the FOI interacts with a surface. As input the previously defined FOI, properties of the surface and optionally further emitters can be used. The result is a new surface property that can be perceived by the sensor. Figure 2 shows an example for the sensor model. It contains the FOI defined in Figure 1, and an additional emitter for fog. To specify the interaction with the surface some of the basic surface properties like normal, reflection and position need to be known. These properties can either be defined as scalar or per point on the surface by using a texture, as usual in shader programming. This allows to model high precise sensor responses like described in [5].

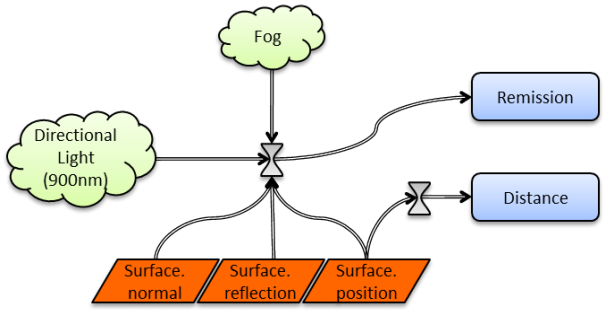

Figure 2. Example Sensor model that represent a simple laser range finder.

C. Sensor Instance Model

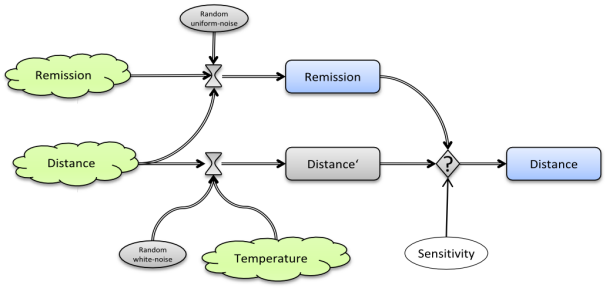

Figure 3. Example Sensor Instance model for a specific laser range finder. Whereas this device is sensitive to temperature fluctuations.

The results calculated in the sensor model, refer to a class of sensors, for example laser range finder and thus are calculated for each instance equally. They can be seen as ideal measurements, without noise or instance specific errors. Within the sensor instance model, a concrete instance of a sensor can be modeled. The previously defined surface properties serve as input. Properties with the same name but a different value define the final result. Like in the model above, further optional FOI, may be used as additional input. In Figure 3 an additional symbol is introduced, the decision between Distance and the final result. In this example it is used to describe the sensitivity of the light receiver of the laser range finder. In other words, if the received light lies below a certain threshold, the measured distance will be set to unknown or zero. Also first used in Figure 3 are random number generators, to model statistical noise. To

 ISBN: 978-1-61208-234-9

cover most of the real noise distribution, random numbers can be generated using different probability distributions and combined if necessary. This post processing step is also intended to model sensor faults, that could follow from an incorrect parameterization or calibration.

D. Computational Model

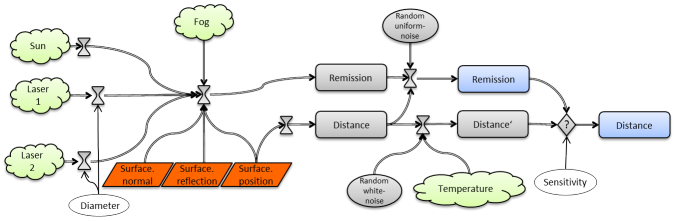

Figure 4. Generated computational model for two active sensors and three emitters of environmental influences.

Figure 4 shows the three previously defined models, summarized within the computational model. The major change is that the emitter (Directional Light in Figure 1) of the emitter model will be replaced by concrete instances of the corresponding class (Laser1, Laser2 and Sun in Figure 4) during combination. In this example, two laser scanners, the sun and fog have an impact on the observed surface. This was determined by the combination of the area observed by the sensor in charge, and the area affected by the emitters. Only those emitters will be considered which have an impact on the perceived surface properties. This is mostly described within the sensor model section.
The computational model is reviewed and newly generated whenever a sensor is intended to record new measurements. Finally the shader code is generated using the computational model in order to enable efficient computation on the graphics hardware. This needs to be done for all objects or surfaces, within the area observed by the current sensor.

IV. Conclusion and Future Work

This paper has dealt with the dynamic simulation of environmental factors and cross interferences between sensors. First, the requirements for creating a simulation environment, with the goal to produce realistic sensor data, has been discussed and compared with currently available simulations. It was found, that in favor of performance, correct influences and error models are often ignored.

In the second part, a method to model environmental influences and sensors has been presented which allows a comfortable way to model realistic sensors. The separation between environmental conditions, sensors and hardware specific errors, helps to incrementally increase the simulation detail until the desired degree of realism is reached.
Since the computational model is reviewed every frame the model can adapt to the currently simulated situation. The usage of shader programs during simulation runtime allows an efficient execution of the computational model which can be interpreted as a system dynamics simulation for each point on objects surfaces.

Acknowledgment

This work has been supported by the Federal Ministry of Economics and Technology (BMWi) under the grant 01MA09037. This text reects the views only of the authors.

References

[1] B. Gerkey, R. Vaughan, and K. Stoy, “Most valuable player: A robot device server for distributed control,” *International Conference on Intelligent Robots and Systems*, no. Iros, pp. 1226–1231, 2001.

[2] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “Ros: an open-source robot operating system,” in *ICRA Workshop on Open Source Software*, 2009.

[3] (2012) The Mobile Robotik Programming Toolkit . Retrieved: July. 2012. [Online]. Available: http://http://www.mrpt.org/

[4] S. Van Hoecke, S. Verstockt, K. Samyn, M. Slembrouck, and R. de Walle, “Tunnel simulator for traffic video detection,” in *SIMUL 2011*. International Academy, Research, and Industry Association (IARIA), 2011, pp. 57–62.

[5] J. Rossmann, N. Hempe, and M. Emde, “New methods of render-supported sensor simulation in modern real-time VR-simulation systems,” in *Proceedings of the 15th WSEAS international conference on Computers*. Stevens Point, Wisconsin, USA: WSEAS, 2011, pp. 358–364.

[6] S. Balakirsky and C. Scrapper, “USARSim : Providing a Framework for Multi-robot Performance Evaluation,” *Simulation*, pp. 98–102.

[7] C. Goodin, P. J. Durst, B. Gates, C. Cummins, and J. Priddy, “High Fidelity Sensor Simulations for the Virtual Autonomous Navigation Environment,” pp. 75–86, 2010.

[8] M. Siegel, “Sensor modeling and simulation: can it pass the Turing test?” *VIMS 2001. 2001 IEEE International Workshop on Virtual and Intelligent Measurement Systems (IEEE Cat. No.01EX447)*, pp. 92–96, 2001.

[9] C. Rosen, U. Jeppsson, L. Rieger, and P. a. Vanrolleghem, “Adding realism to simulated sensors and actuators.” *Water science and technology*, vol. 57, no. 3, pp. 337–44, Jan. 2008.

[10] C. Goodin, R. Kala, A. Carrrillo, and L. Y. Liu, “Sensor modeling for the Virtual Autonomous Navigation Environment,” *2009 IEEE Sensors*, pp. 1588–1592, Oct. 2009.

[11] K. Khoshelham and S. O. Elberink, “Accuracy and resolution of Kinect depth data for indoor mapping applications.” *Sensors (Basel, Switzerland)*, vol. 12, no. 2, pp. 1437–54, Jan. 2012.

[12] (2012) OGRE. Retrieved: Sept. 2012. [Online]. Available: http://www.ogre3d.org/

[13] F. Ford, *Modeling the environment: an introduction to system dynamics models of environmental systems*. Island Pr, 1999.

 ISBN: 978-1-61208-234-9