National Physical Laboratory

NPL REPORT MS 29

A SENSOR MODELLING FRAMEWORK FOR AUTONOMOUS SYSTEMS

RAHUL KHATRY AND ANDREW THOMPSON

APRIL 2021

Department for Business, Energy & Industrial Strategy

FUNDED BY BEIS

A sensor modelling framework for autonomous systems

Rahul Khatry and Andrew Thompson
Data Science

ABSTRACT

Sensor modelling is an important aspect of developing and testing automated vehicles. This document describes a standardised framework for the practice of developing such models. We describe a general approach, illustrating with examples for different sensors and sensor models.

ISSN 1754-2960

https://doi.org/10.47120/npl.MS29

National Physical Laboratory
Hampton Road, Teddington, Middlesex, TW11 0LW

This work was funded by the UK Government's Department for Business, Energy and Industrial Strategy (BEIS) through the UK's National Measurement System programmes

Approved on behalf of NPLML by
Louise Wright, Head of Science (Data Science).

CONTENTS

1 Introduction

Sensor modelling is an important building block for the simulation of automated vehicles. The role of a sensor model is to mimic the process of capturing information from the environment and providing it for the subsequent processing steps. Sensor models in the context of automated vehicles have been developed for cameras [1], LiDAR [2], RaDAR [3], Sonar [4], and other sensor types, and modelling each specific sensor type is an in-depth area of study in itself. While lots of sensor models have been developed, there has not been a significant effort in standardising the process of sensor modelling and this work is step in that direction. The present document describes a general framework for creating sensor models which is applicable to all the types of sensor mentioned above. With this type of standardisation we hope to eventually achieve an informed process of sensor model development keeping rigorous coverage of ODD, safety standards and uncertainty quantification and reporting in mind.

The document is structured as follows. Section 2 motivates the concept of a sensor model and gives an overview of some common types of sensor model. Section 3 then describes an overall workflow for the creation of a sensor model. This would be the first attempt to standardise the process of modelling different sensors in automated vehicles and hopefully serves as a building block towards greater standardization in virtual testing. We illustrate each step in the context of different kinds of sensor models.

2 Sensor Models

In the field of automated vehicles, physical testing is a cumbersome and costly process and cannot be done for every type of scenario that a vehicle might encounter, and hence virtual testing becomes very important. In order to perform virtual testing, many kinds of testing architectures can be considered including Vehicle-In-the-Loop (MIL), Software-In-the-Loop (SIL), and Hardware-In-the-Loop (HIL), see [1] for more details. ISO 26262 [2] provides some guidance on the use of simulation in verification and validation activities and is based on the V-model. At the heart of performing all these virtual tests is the presence of a reliable sensor model consisting of rigorous coverage of operational design domain and proper understanding and reporting of accuracy and uncertainty.

A sensor model is a mathematical construct designed to mimic the workings of a sensor in the real world. Sensor models are used in many applications, including automated vehicles and healthcare [3], [4]. Sensor models also function as the link between the running system (automated vehicle driving algorithms etc) and the virtual environment which is important for situational awareness.

A model is a function that takes in a set of inputs and mathematically generates a set of outputs, and it is important to be clear what these inputs and outputs are. The input to a sensor model for this application consists of two parts: the scenario and the environmental conditions (for example rain). For the model to be computationally tractable, both inputs and outputs need to be discretised. For example, for a camera model, the input scene and the output would most likely be a pixelated grid where each pixel captures the average intensity of the incident light within its boundary. The environmental conditions would need to be summarized by some finite dimensional feature vector [5]. For LiDAR, RaDAR and Sonar, on the other hand, while the environmental conditions would need to be captured similarly, the input scene would most likely be an occupancy grid (a spatial map of the AV's environment) [6] and the outputs would be the intensity and range measurements recorded by the sensor. In this context, the inverse sensor model is also of interest, which solves for the occupancy grid given the LiDAR, RaDAR or Sonar measurements [7]. The forward sensor model [6] gives the conditional probability p of observing a reading z_k when the state of the i^{th} cell is m_i, namely

$$p(z_k \mid m_i),$$

whereas an inverse sensor model gives the probability that the state of the i^{th} cell is m_i given a reading z_k, namely

$$p(\, m_i \mid z_k).$$

An example of an occupancy grid can be seen in Figure 1.

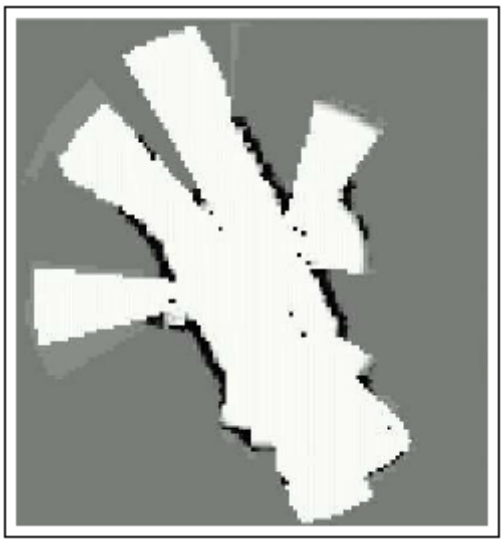

Figure 1 A typical occupancy grid for robot navigation [6]

The sensor model should also capture as closely as possible the different effects of environmental conditions. Figure 2 shows the contributions towards the functionality of different kinds of sensors. Conditions such as rain, snow, fog or dusty environments affect different sensors in different ways. LiDAR for example is strongly affected by rain whereas RaDAR is less affected by rain. Cameras perform poorly under low lighting, whereas LiDAR and RaDAR are less affected by lighting conditions [3].

Having established the inputs and outputs of the model, we next consider the creation of the sensor model.

3 Sensor Modelling Framework

This section presents a generic framework for creating sensor models. The different steps of the workflow are illustrated in Figure 3. The first step is the scoping step in which we determine which aspects the sensor model should take into account. The next steps are to decide on the data to capture the appropriate phenomena and to decide on the model that would be appropriate for capturing the necessary effects. Once the model and data are decided, the next steps are to choose the most appropriate method for fitting or training the model over the given data, and to choose an appropriate method for uncertainty quantification such that the systematic and random effects can be accounted for and propagated through the model. The final steps are to fit/train the model and evaluate the model performance.

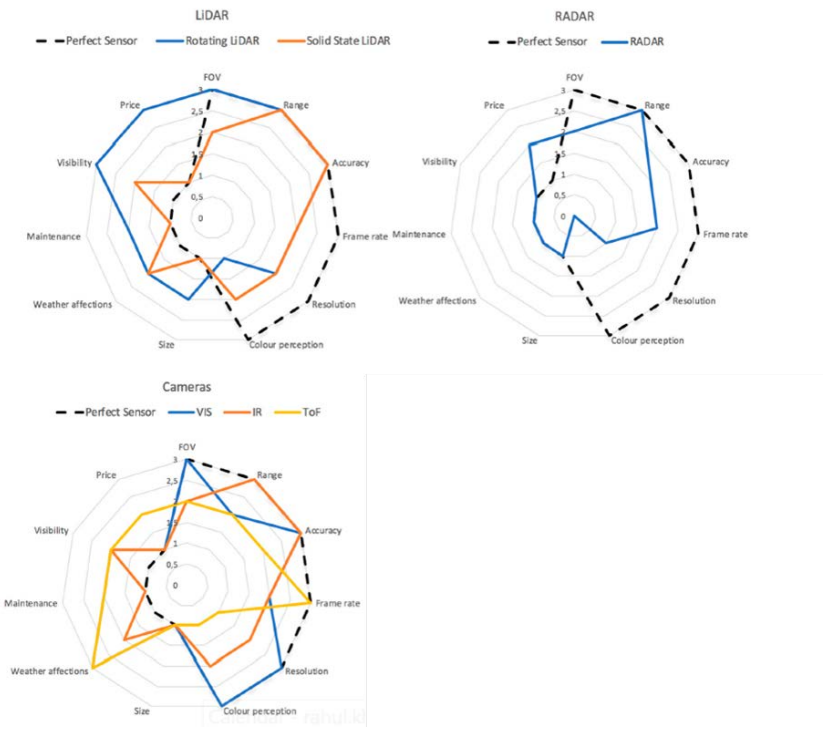

Figure 2 Sensor capability spider diagram [3]

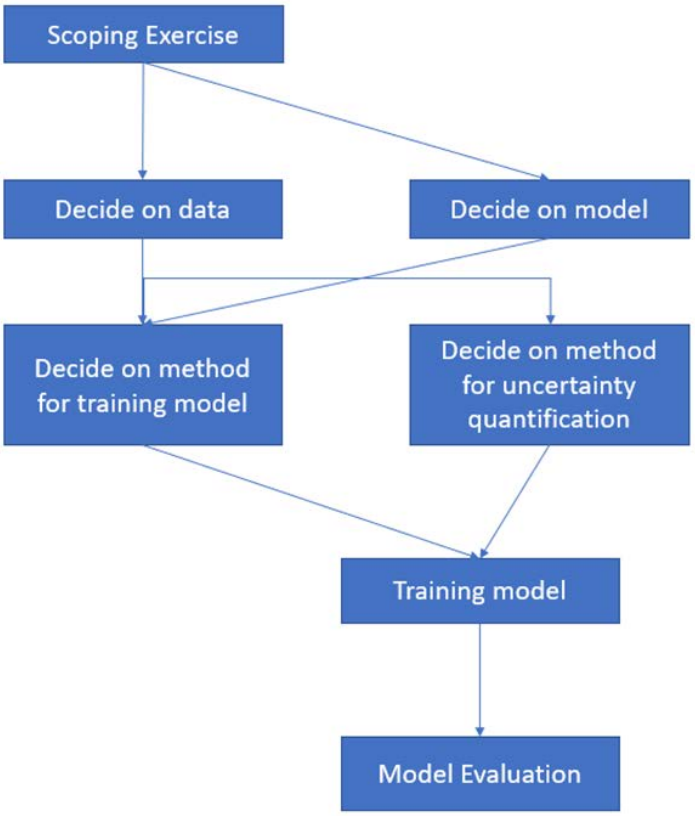

Figure 3 Sensor modelling framework flowchart

3.1 Scoping exercise

The first steps are to understand the scope of the model and to specify the different phenomena that need to be captured by it. Taking the example of a camera, this scoping exercise would identify phenomena affecting the acquisition process such as chromatic aberration, motion blur, exposure automation and lens reflection [5].

The complexity of the model should be sufficient that it is able to characterize the different key performance indicators (KPIs) of the sensor so that the model predictions and the sensor performance can be compared, and so that the key performance indicators can be calculated for scenarios of interest. Some possible key performance indicators appropriate for sensors on autonomous vehicles are listed below, though some are specific to particular types of sensor.

i. Angular/spatial resolution: capability to discriminate between two adjacent targets
ii. Range resolution: capability to discriminate between two targets at different distances
iii. Speed resolution: capability to discriminate between two targets at different speeds
iv. Maximum detectable range: the maximum range at which a target can be detected
v. Minimum detectable range: the minimum range at which a target can be detected
vi. Maximum unambiguous speed: the maximum speed that can be measured unambiguously
vii. Minimum detectable speed: the minimum speed that can be measured
viii. Update rate/responsiveness: the speed with which the systems provides new outputs
ix. Tracking capability: the capability of the system to target/targets in the scene
x. Tracking capacity: the number of targets that the system can track at the same time
xi. Contrast: the variance in luminance between the target and the scene background
xii. Sidelobe levels: the ratio between the main lobe and the first sidelobe peak in the ambiguity function
xiii. Integrated Sidelobe Ratio: the ratio between the main lobe power and the sum of the power of all the sidelobes in the ambiguity function
xiv. False Alarm Probability: the probability of detecting a target when it is not present
xv. Detection probability: the probability of detecting a target when it is present
xvi. Dynamic range: the ratio between the maximum and minimum values of the received signal power that the system can measure
xvii. Noise floor: the signal consisting of the sum of all the noise sources and unwanted signals measured by the sensor
xviii. Linear range: the range in which the output of systems applies a linear function to the input
xix. Linearity: the deviation of the measured response curve from an ideal straight line
xx. Antenna patterns: the radiation pattern of the systems sensor antennae
xxi. Field of view: the angle through which the system can detect electromagnetic radiation

The KPIs will help identify the range of scenarios that are of interest to be modelled, and this will also inform the decisions about data collection. The scoping exercise should also identify which environmental effects are relevant for the given sensor (using physical insight and expert knowledge). Relevant ranges of the environmental effect variables also need to be identified.

3.2 Deciding on the model

The next step is to design a model that can capture the relevant phenomena. A flow chart is likely to be helpful for capturing which effects are to be included and the order in which they are to be considered in the model, and Figure 4 gives an example of what such a flow chart might look like in the case of a camera [5].

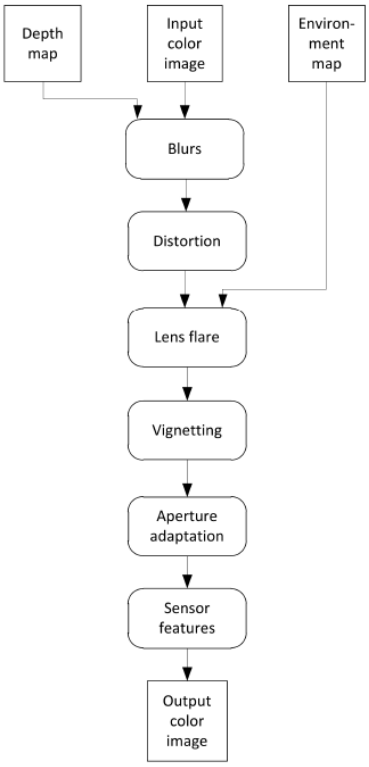

Figure 4 Camera sensor model flow chart [5]

The next crucial decision is whether to use a model that is informed by the underlying physics (a physics-based model) or not (a data-driven model). This decision will largely be influenced by two issues: how well is the physics behind the model understood, and how much data is available? If the physics is well understood and if information concerning model parameters (for example lens characteristics) is available, a physics-based model is likely to be more accurate and more straight forward to fit. It is likely, however, that information concerning model parameters is not readily available in an autonomous vehicle setting, especially if the sensor has been developed by someone else. If the underlying physics is not well understood, a data-driven model may be a necessity. The requirements of data coverage and quality typically mean that a large and carefully collected dataset is required in order to train a data-driven model. Data may also be needed in order to fit the parameters of a physics-based model, but the data requirements are typically less stringent due to the information contained and constraints imposed by the known model.

There may be other considerations which impact the choice between physics-based and data-driven models. If an additional requirement of the model to provide insight into why certain outputs are observed, only a physics-based model is able to do this. From a different point of view, accompanying model outputs with uncertainty evaluation is more straightforward for models which are simpler to express mathematically, and sometimes accuracy of data-driven models can only be achieved by considering mathematically complex models such as deep neural networks. For a given sensor, a physics-based model may be more appropriate for some parts of the model and a data-driven model may be more appropriate for other parts, leading to a hybrid model.

3.2.1 PHYSICS-BASED MODELS

We next give some examples of physics-based models.

(i) Lens flare and aperture adaptation for cameras.

Given an input image S, the output image B after lens flare [5] may be modelled as the convolution of S with a fixed convolutional kernel K, namely

$$B = S * K.$$

We note that the convolutional kernel K here represents a parameter of the model which needs to be determined. Similarly, for an input image S, the output image B after aperture adaptation [5] can be modelled simplify as an image-wide multiplication by some constant, namely

$$B = c\left(\frac{ts}{f^2}\right) S,$$

where c is a constant, t is exposure time, s is the ISO film speed and f is the f-number. The parameters t, s and f can usually be found in the data provided by the camera manufacturer, and the constant c would need to be determined through experimentation.

(ii) Occupancy map modelling for RaDAR, LiDAR and Sonar.

In RaDAR, LiDAR and Sonar, each reading consists of an estimated distance to the nearest obstacle within the cone of the sensor. However, obstacles are not guaranteed to reflect rays back to the sensor, but instead can be modelled as doing so with a certain probability. Given an occupancy map of the scene, a probabilistic model of a single reading can be used. For example, in [6] a model is considered in which the probability of a range reading corresponding to an occupied cell is distributed according to a geometric distribution (since it is most likely that the nearest obstacle is detected) convolved with a Gaussian (since the reading is noisy).

The inverse map problem is then to reconstruct the occupancy map (or point cloud), and hence the positions of real objects, from a series of such readings. The general approach is to use Bayesian inference to infer the state of each cell given the measurements. This can either be done cell-by-cell using a standard application of Bayes' Rule, or for the whole grid together by considering the joint likelihood function over all cells and using the expectation-maximisation (EM) algorithm to solve for the maximum likelihood estimator [6].

The diagram in Figure 5 taken from [8] illustrates the building of an occupancy grid map for LiDAR sensors, highlighting the limitations of cell-by-cell inversion.

3.2.2 DATA-DRIVEN MODELS

Turning to data-driven models, many different types of model can be considered. This makes the approach extremely flexible, but it also means that thought must be given to the choice of model. Questions to be asked include:

(i) Is a linear or nonlinear relationship expected between inputs and outputs?
(ii) If nonlinear, what type of mathematical relationship might be expected?
(iii) Does the dependence upon a given variable have different regimes, for example a linear region and a nonlinear region?

It is often helpful to think of models in terms of where they sit on the complexity spectrum. The complexity of data-driven models can range from simple linear models to highly complex deep neural networks, and the complexity of the model should be chosen to match expectations.

Some of these expectations might emerge from the scoping exercise, but they may also need to be refined through an iterative process of fitting trial models on initial data (see Section 3.3). Sensor outputs may also have a more complex dependence upon some inputs compared to others, and the model may need to be chosen to reflect this difference.

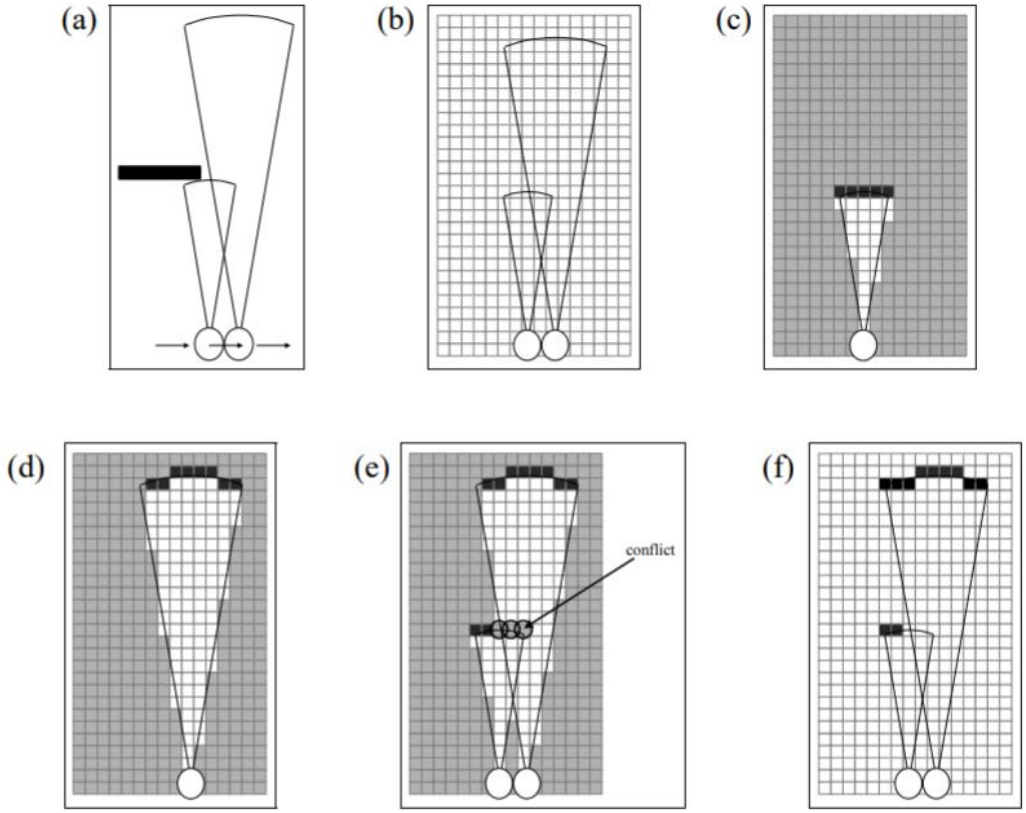

Figure 4 Occupancy grid generation using a sonar sensor [6]

Figure 5 illustrates the generation of occupancy grid points. A passing robot receives the range free measurement in (a), the map is divided into grid cells as in (b) and the inverse sensor model maps the beam into probabilistic maps which is done separately for each grid cell as shown in (c) and (d). These are then superimposed to obtain the map shown in (e) and then a map that explains the conflicts is obtained like that in (f).

A simple example of a data-driven model is polynomial regression, where the outputs are assumed to be a polynomial function (of a certain degree) of the inputs. Writing $\mathbf{x} = \{x_i, i = 1, 2, \ldots, N\}$ for the input vector and writing $f(x)$ for a (univariate) output of the model, a quadratic (degree 2) model of $f(x)$ would take the form

$$f(\mathbf{x}) = a + \sum_{n=1}^{N} b_n x_n + \sum_{n,m=1}^{N} c_{nm}\, x_n x_m$$

3.3 Data collection

The third step is to decide on the data that needs to be captured in order to fit the model to the sensor.

Methods of capturing the necessary data for cameras may include capturing images from a calibrated camera sensor with various distortion effects whose parameters are known. Real data could be captured by experimental vehicles equipped with the necessary sensors, and appropriate post-processing. This could be a time consuming and expensive process since to capture enough images to cover all different kinds of scenarios would be a big task, and hence it may make sense to generate images by simulating scenarios of interest [10]. Variations in scene intensity, pixel size, exposure duration, and camera motion would need to be present in the captured dataset.

For LiDAR, RaDAR and Sonar, data capture could amount to generation of scan grids or occupancy grids. Scan grids are single shot recordings of occupancy grids generated from a single point cloud, whereas occupancy grids build up the scene over time by accumulating scan grids. The measurement campaign could also consider objects with different reflection and scattering properties, to ensure that the variety of those properties seen in objects to be sensed in the real world is captured by the model.

The following points should be considered when deciding what data it is sufficient to collect to model the behaviour of a sensor.

i. **Complexity**. What is the complexity of the model (see Section 3.2)? In general terms, the more complex the model, the more data is required to fit the model (and also to validate it – see Section 3.7). As was noted in Section 3.2, sensor outputs may also have a more complex dependence upon some inputs compared to others. A more dense sampling of input variables whose relationship with outputs is more complex will be required. While the complexity of a physics-based model might be high, much of the complexity has already been captured in the form of the model and it remains to learn the parameters, and in this case the data requirements are much less stringent.
ii. **Smooth coverage**. Does the data set provide smooth coverage of the output space? If there are regions of large changes in the output values that do not contain many data points, or if there are large changes in output values between points that are close in the input space, more data may be required. However, it may be that response to some of the inputs in truth exhibits discontinuities. Smooth coverage of the output space at these discontinuities is unrealistic.
iii. **Sensitivity**. What is the sensitivity of the response of the sensor to each of the inputs? Do some of the inputs affect the sensor response significantly more than others? Input variables to which the outputs are more sensitive should be sampled more densely. On the other hand, the sensor may be found to be invariant to some of the inputs, and in this case these variables can be removed from the model.
iv. **Limits**. How does the sensor response behave at very low and very high levels of the variable quantities?

3.4 Deciding on the method of training/fitting

The next step is to decide on methods for training the model, or in other words to fit the parameters of the model in order to capture the behaviour of the sensor as closely as possible.

For physics-based models, the aim is typically to perform data-driven estimation of a small number of parameters. A popular approach to this problem is nonlinear least squares, in which the sum of squared differences between the measurement outputs and those of the model is minimised to obtain estimates of the parameter values. If derivatives with respect to the parameters can be determined then gradient methods can be used to solve the optimisation problem, and if not a global optimisation method would need to be used.

Returning to the two camera examples from Section 3.2, stochastic gradient descent is often used to estimate the kernel K in the lens flair model [5], while approximating c in the aperture adaptation formula can be done using an extremely simple one-dimensional least squares fit.

For data-driven models, there are straightforward and well-established methods for fitting some of the less complex models. Taking the example of polynomial regression from Section 3.2, given input data $\{x^p\}$ and paired output data $\{y^p\}$, one approach to fitting this model is to minimize the error between the measured output data y^p and the fitted predictions $f(x^p)$. For example, the (nonlinear) least squares approach would mean minimizing

$$\sum_{m=1}^{M} \left(y^p - f(x^p)\right)^2$$

with respect to the parameters. An alternative approach is to introduce additional variables for the higher-order polynomial terms and solve using linear least squares.

At the other end of the spectrum, the parameters (usually called weights) of a neural network are typically optimized using variants of batch gradient descent, for example the Adam optimizer, and these algorithms are easily called in popular deep learning environments such as Tensorflow. Neural networks are usually highly overparametrised, and while this means they are highly descriptive, care must be taken not to overfit the training data in a way which doesn't generalise, see [10] for more details on deep learning.

3.5 Deciding on the method of uncertainty quantification

Measurement uncertainty is a statement of confidence in the value of a measured quantity. Uncertainty can never be eliminated altogether, and therefore no model output is complete without an accompanying statement concerning its uncertainty. Several sources of model output uncertainty can be distinguished [11].

(i) For any model, **uncertainties in the inputs to the model** will propagate through the model to the outputs. For a sensor, these inputs include the scene, the acquisition process and the environment (random fluctuations in weather conditions for example).

(ii) There may also be **uncertainty concerning the parameters of the model**. In a physics-based model, if the parameters are set using prior knowledge, uncertainties capture strength of belief in the selected values. If the model is trained using data, parameter uncertainty can be caused by training data with insufficient coverage. For models where parameter optimisation is a nonconvex problem, it is difficult to be sure that the parameters obtained are globally optimal. Uncertainties in the training data itself, both inputs and outputs, also contribute to uncertainty in the fitted model parameters.

(iii) Model uncertainty quantification usually depends upon the assumption that, subject to optimal parameter fitting, the chosen family of models contains the correct model. If a data driven model has insufficient complexity, this assumption may not be valid. If this contribution to uncertainty is ignored, uncertainty estimates are unlikely to be reliable. Alternatively, **model inadequacy** can be viewed as a systematic effect, contributing additional uncertainty.

(iv) Uncertainty also arises due to **numerical errors and numerical approximations** in the implementation of the model. These uncertainties are usually not dominant and are typically ignored.

Obtaining parameter uncertainties for data-driven models can be posed as a Bayesian inference problem, where prior belief concerning parameters is combined with available data to estimate the posterior distribution of the parameters. In some simple cases, this approach has closed-form solutions, for example linear regression with i.i.d. Gaussian noise [12] and Gaussian processes [13], and so uncertainties come essentially 'for free' with the model fitting. Often, however, sampling methods such as Monte Carlo are needed to approximate the posterior distribution of the parameters. Bayesian inference is especially challenging for neural networks due to the large number of parameters and their highly overparametrised nature, which makes it difficult to assign prior belief to the parameters [10].

Knowledge concerning input and output uncertainties can also be incorporated into the model and consequently into the Bayesian inference. Standard approaches are able to deal with uncorrelated output uncertainties, but the problem becomes more challenging if the output uncertainties are correlated (which is likely to be the case if there is systematic model misfit – see point (iii) above). Incorporating input uncertainties into Bayesian inference is also a challenging problem, addressed for example by errors-in-variables approaches.

Propagation of uncertainty through sensor models generally requires a numerical approach rather than the analytical approach recommended by the Guide to the Expression of Uncertainty in Measurement [14], because the relevant analytical information is not generally available. The most general numerical approach, Monte Carlo sampling (MCS), is described in full in GUM Supplement 1 [15]. It assembles a cumulative density function for the output of interest by repeatedly evaluating the model with randomly chosen values of the inputs. For a reliable evaluation of uncertainty MCS requires thousands of model evaluations, which may be a problem if the model is computationally expensive. Alternative sampling methods can reduce this expense, as can the use of surrogate models. Both of these options are discussed in more detail elsewhere [16].

3.6 Training the model

Once the data has been collected and the model and method of training/fitting has been decided upon, the training/fitting of the gathered data to the chosen model can be performed to obtain a working model. Sometimes training is not a one time activity, but instead the training is carried out online with the aim of improving the model over time as the model is given more scenarios and data.

3.7 Validating the model

Validation of a model means performing testing to judge how accurately it approximates the behaviour of the real sensor. In the context of autonomous vehicles, there are often a wide range of potential scenarios in which the model needs to accurately reflect reality, and so validation involves testing the model thoroughly against the full range of possible scenarios. It is generally far too time consuming to collect real world data covering the full range of scenarios, and hence simulation-based virtual testing approaches such as Software-In-the-Loop (SIL) and Hardware-In-the-Loop (HIL) become useful as a way of reducing the testing time and cost [1].

Different scenarios can be generated based upon what the operational design domain of the system needs to be. For example, an automated vehicle may be exposed to different kinds of

lighting and weather conditions and may need to identify and spot obstacles of different types, reflectivities, sizes, colours, poses, etc. The sensor model thus needs to be exposed to such variation and tested as to whether the model is performing accurately in all those conditions. For occupancy grid methods, experimental data and model generated data can be directly compared either by visual inspection or using quantitative metrics such as Baron's correlation coefficient and Pearson's correlation coefficient [9].

Virtual testing for safety certification will require careful definition of the scenarios to be tested. Different scenarios are likely to be important in different geographical areas due to the variations in weather, infrastructure, driving culture, wildlife, and so on. It is also important to note that a completely different way of thinking about unsafe scenarios is necessary, because an AI system has a much narrower field of experience than a human and has a completely different set of sensory information. This all makes validation of sensor models in an autonomous vehicle setting a challenging task, requiring a great deal of work to decide the appropriate tests and evaluate in which scenarios the model is performing well and in which scenarios it is not.

4 Conclusion

We have described a general framework for creating sensor models for autonomous systems. Sensor models are needed in order to simulate sensors' working both in normal conditions and upon edge cases within a virtual testing environment. The models developed here can generate edge cases within a virtual testing environment so that a vehicle AI is presented with realistically distorted images during testing and so that dangerous edge cases can be checked without creating physically dangerous real-world situations. A companion report [20] demonstrates how to model certain aspects of the functioning of cameras and Lidar sensors, using the sensor modelling framework to construct the models, addressing several edge cases in the process.

The ultimate aim of this work is to support the creation of standardised virtual testing frameworks within the autonomous transport industry. Such standardisation across the industry could result in a common language which would improve collaborative research and result in the development of high-quality testing standards and improved communication between industry and suppliers. One important piece in this puzzle is standardising approaches to sensor modelling, and the work described here is a first step in this direction.

Future work will now focus upon engaging stakeholders in order to gain a more informed understanding of the types of weather conditions, edge cases and distortion phenomena which are most relevant for the various sensors in the autonomous vehicle sensing stack. Eventually with such standardization in sensor modelling and interfacing, virtual testing can get one step closer to being adopted across the industry as a part of safety testing or type approval in automated vehicles. This would also be a step towards improvement in uncertainty reporting and estimation in automated driving systems which is an important component of safe operation in any system.

5 References

[1] WuLing Huang et al. Autonomous vehicles testing methods review. In IEEE 19th International Conference on Intelligent Transportation Systems (ITSC) (pp. 163-168), IEEE, 2016.

[2] https://www.iso.org/obp/ui/#iso:std:iso:26262:-9:ed-1:v1:en, ISO26262 V-Model

[3] Francisca Rosique et al. A systematic review of perception system and simulators for autonomous vehicles research. Sensors 19(3), 2019.

[4] Stefanie Pöhlmann et al. Evaluation of Kinect 3D sensor for healthcare imaging. Journal of medical and biological engineering 36(6), 2016.

[5] Michal Kucis and Pavel Zemcki. Simulation of Camera Features. Proceedings of CESCG: The 16th Central European Seminar on Computer Graphics, 2012.

[6] Sebastian Thrun. Learning occupancy grid maps with forward sensor models. Autonomous Robots 15(2), 2003.

[7] Rob Weston et al. Probably unknown: Deep inverse sensor modelling radar. In 2019 International Conference on Robotics and Automation (ICRA) (pp. 5446-5452), IEEE, 2019.

[8] Kaustubh Pathak et al. 3D forward sensor modeling and application to occupancy grid based sensor fusion. In IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 2059-2064), IEEE, 2007.

[9] Alexander Schaermann and Timo Hanke. BMW Group: Generation and Validation of Sensor Models for Automated Driving Systems Using VIRES VTD, Engineering Reality Magazine, Volume IX, 2019.

[10] Ian Goodfellow, Yoshua Bengio and Aaron Courville. Deep learning. Vol. 1. Cambridge: MIT press, 2016.

[11] Pradeep Ramuhalli et al. Uncertainty Quantification Techniques for Sensor Calibration Monitoring in Nuclear Power Plants, Prepared for the U.S. Department of Energy, 2014.

[12] Jerome Friedman, Trevor Hastie and Robert Tibshirani. The elements of statistical learning. Vol. 1, No. 10. New York: Springer Series in Statistics, 2001.

[13] Christopher Williams and Carl Rasmussen. Gaussian processes for machine learning. Vol. 2, No. 3. Cambridge, MA: MIT press, 2006.

[14] BIPM, IEC, IFCC, ILAC, ISO, IUPAC, IUPAP and OIML. Evaluation of measurement data - Guide to the expression of uncertainty in measurement. Joint Committee for Guides in Metrology, JCGM 100:2008, 2008.

[15] BIPM, IEC, IFCC, ILAC, ISO, IUPAC, IUPAP and OIML. Evaluation of measurement data – Supplement 1 to the Guide to the expression of uncertainty in measurement - Propagation of distributions using a Monte Carlo method. Joint Committee for Guides in Metrology, JCGM 101:2008, 2008.

[16] Knud Rasmussen et al. Novel Mathematical and Statistical Approaches to Uncertainty Evaluation: Best Practice Guide to Uncertainty Evaluation for Computationally Expensive Models. Available from http://www.mathmet.org/publications/guides/index.php#expensive.

[17] https://techcrunch.com/2020/05/06/volvo-to-use-luminars-lidar-in-production-vehicles-to-unlock-automated-driving-on-highways

[18] https://blog.waymo.com/2020/03/introducing-5th-generation-waymo-driver.html

[19] https://www.therobotreport.com/researchers-back-teslas-non-lidar-approach-to-self-driving-cars/

[20] Rahul Khatry and Andrew Thompson (2021). Camera and Lidar sensor models for autonomous vehicles. NPL Technical Report (draft), available upon request.