Journal of Software Engineering for Robotics
8(1), December 2017, 17-31
ISSN: 2035-3928

Declarative Rule-based Safety for Robotic Perception Systems

Johann Thor Mogensen Ingibergsson*† Dirk Kraft* Ulrik Pagh Schultz*
* University of Southern Denmark, Campusvej 55, 5230 Odense M, Denmark
† CLAAS E-Systems, Møllevej 11, 2990 Nivå, Denmark

Abstract—Mobile robots are used across many domains from personal care to agriculture. Working in dynamic open-ended environments puts high constraints on the robot perception system, which is critical for the safety of the system as a whole. To achieve the required safety levels the perception system needs to be certified, but no specific standards exist for computer vision systems, and the concept of safe vision systems remains largely unexplored. In this paper we present a novel domain-specific language that allows the programmer to express image quality detection rules for enforcing safety constraints. The language allows developers to increase trustworthiness in the robot perception system, which we argue would increase compliance with safety standards. We demonstrate the usage of the language to improve reliability in a perception pipeline and evaluate it against manually written rules on embedded hardware. The language allows the vision expert to concisely express the safety-related constraints and thereby bridging the gap between domain experts and certification authorities.

Index Terms—DSL, readability, safety, functional safety, computer vision

1 Introduction

Mobile robots are used across many domains from personal care to agriculture. For domains such as agriculture where heavy machinery is used extensively, robots can improve performance and efficiency while increasing safety, which is critical even for developed countries [1]. Despite the increased development of mobile robots, designing robots remains a difficult task, in part due to the many overlapping domains. We are interested in the *field robotics* subdomain of mobile robots, which concerns machinery applied for outdoor tasks, e.g., in construction, forestry and agriculture [2], [3]. The development of field robots is particular demanding, because the robots operate in an open-ended dynamical environment [4], which introduces additional strain on the robots, and as result outdoor mobile robots fail up to 10 times more often than other types of robots [5].

The reliability issue for robotics has been discussed in terms of systematic software framework design to improve quality [6]. We however observe that the same issue of reliability has already been addressed in more mature domains, such as the avionics and automotive domains, for example using Model-Driven Engineering (MDE; [7]). MDE is similarly being used in robotics to improve development time and reliability [8], [9]. These MDE approaches are however not developed with a functional safety focus and are missing some key aspects to make the robots trustworthy in relation to certification and consumers.

Safety certification is a method for achieving industry-required levels of reliability and dependability, while addressing liability (see [10]). Compliance with safety standards is considered key to ensuring reliability by achieving an appropriate level of functional safety [11]. We therefore believe that MDE is an important method for achieving compliance. In order to ensure safe autonomous operation, robust and reliable risk detection and obstacle avoidance must be performed. In this regard, field robots are highly dependent on perception sensors and algorithms to understand and react to the environment. The robot has to observe a large area; it must be fast, reliable and robust; it is constrained to function with low computational resources due to embedded hardware; and it might have lower priority than control; and must be predictable [12]. This imposes severe constraints on the software that interacts with sensors.

We propose to use a Domain Specific Language (DSL) for safety concerns in perception systems. We see code generation

Regular paper – *Manuscript received July 15, 2017; revised Nov 15, 2017.*
Digital Object Identifier: 10.6092/JOSER_2017_08_01_p17

as an improvement to reliability and the possibility of lowering the demands for achieving certification, as is done by Bensalem et al. [13]. Our current research question is to investigate the applicability of safety rules for perception systems, based on code generation from a DSL. Concretely our goal is to investigate if it is possible to create rules for sensory systems, as has been done with control [14]. Existing MDE methods describe safety issues, however these issues are not addressed according to any standards [3]. Instead focus has been on quality; standards within software for field robots are only used to a very limited extent and not existing within computer vision [3].

Paper Outline: First, Section 2 outlines background and related work on safety, computer vision, and MDE, using the related domain of automotive as a concrete case, followed by Section 3 which presents the main contribution our DSL. The syntax and semantics is presented along with the current state of implementation. We then evaluate the DSL quantitatively and qualitatively, leading to an overall discussion of the use of the DSL. Last, Section 7 concludes and describes future work.

Previous Work: This is an extended version of a paper originally published at IEEE IRC [15]. Compared to the original version, we have expanded related work with more detail; the description of the language has been extended, by more in-depth information along with syntax and semantics; and we have evaluated new concepts and benchmarked verified computer vision rules on embedded hardware, along with a qualitative analysis of the language.

2 Background

This section introduces standards and functional safety in the agricultural domain, leading to safety in computer vision and the connection to the automotive domain, ending with an overview of MDE methods.

2.1 Functional Safety

Safety as defined by the International Organization for Standardization (ISO) is the ability to mitigate identified risks, i.e. detected hazards resulting in harm, where risk is the "*combination of the probability of occurrence of harm and the severity of that harm*" [16] and harm is defined as "*injury or damage to the health of people, or damage to property or the environment*" [16]. We deal with safety in this sense, which can be summed up by the definition by Avižienis et al., namely the absence of catastrophic consequences [17]. Functional safety is a means for guaranteeing reliability towards specific errors in the entire system (hardware components, hardware design, and software) or as a guaranteed reaction, i.e., reaching a safe state. To comply with functional safety is a matter of conducting a Hazard and Risk Analysis (HRA), where hazards are errors or failures within the system components where a malfunction will result in a dangerous situation. These functions are called safety-related functions. A safety-related function could be detecting obstacles, where a hazard is that the image is overexposed, resulting in missed detections. The HRA then imposes requirements on the entire development process, through the formulation of safety goals. A safety goal, in this case, would ensure that the image exposure is usable for the algorithms. The HRA can result in multiple safety goals, where each goal will result in the development of safety function(s), that ensure or monitor that the safety goal will not be violated [18]. Functional safety focuses on mitigating unacceptable risks, by removing them or ensuring that the system can return to a safe state. The standards, therefore, do not care about nominal performance. Meaning during certification nominal performance is not assessed with respect to functional safety standards. This, however, does not mean that nominal performance is not critical for the system as a whole and therefore should be guaranteed through testing, but should be seen with respect to performance standards or internal quality requirements instead of functional safety.

The mobile robotic domain is only sparsely covered by standards. One standard that comes close is ISO 13482 for personal care robots [19]. This standard has type descriptions for robots where one is categorised as outdoors. The standard impose requirements for the HRA, but relies on other standards for the development requirements. Standards like ISO 13482 have an overview of hazards which needs to be addressed for mobile robots; an extension to this based on the material for ISO 13482 can be found in Dogramdzi et al. [20], which includes descriptions of how to create the HRA for a robot system. Because we are focussing on the sub-domain of field robots we believe that the agricultural functional safety standard (ISO 25119 [21]) should be used for the development. To improve the safety of field robots in general, we believe it is important to give researchers an easy way of integrating these standards into development [3].

Functional safety standards only address human dangers, e.g., ISO 26262 [22] and ISO 25119 [21]. This leaves the designer and developer to categorise issues related to harming the robot, e.g., untraversable ground and non-human obstacles. Hedenberg et al. [23] use EN 1525 (driver-less trucks [24]), and argue to also take into account damages occurring by colliding with an object that indirectly harms people, as well as if material damages resulting from a crash if they are high. That is however much broader than the international standards, nevertheless the same issue exists in European law: loss can be both economic and non-economic; it includes loss of income or profit, burdens incurred and a reduction in the value of property; and also physical pain and suffering and impairment of the quality of life [10]. Overall this means that safety should be addressed for the entire operation of the robot. The safety issue is partially at odds to how autonomy is developed today. Researchers rely on machine learning and Artificial Intelligence (AI) to achieve required performance and safety. The issue however is that for a robot to be perceived safe its behaviour should be understandable for humans to allow for certification. This tension is described in "Defense Science Board 2016 Summer Study on Autonomy" which argues that it is hard for humans to understand and predict AI systems [25].

2.2 Safety and Computer Vision

In practice autonomy needs to rely on perception to allow the robot to operate in a dynamic environment, making the vision domain critical for the safety of the robot. This is also seen in the automotive domain, since if "applications are based on unreliable data [this] causes them to be unreliable as well" [26]. For the automotive domain the challenges of limited hardware resources, low energy consumption, high reliability and robustness and, adverse environmental conditions, must all be addressed to enable computer vision and advanced software to be applicable in real-world scenarios [27]; we expect that this will also be the case for field robots. We therefore look at the robotic domain but utilise information from the automotive industry for concerns and hazards in Section 2.2.1. We use the automotive domain for experience as we look at possible failures when investigating autonomy for cars in Section 2.2.2, because similar failures could be present in robots.

2.2.1 Field Robotics and Automotive Industry

To achieve safety, field robots need to be dependable, which imposes high requirements on the reliability of the software, data and the vision pipeline. Despite the need for safe and dependable computer vision systems, the domain is solution-driven [3]. There do exist probabilistic measures for failure detection [28], [29], the issue with these methods is however that it is difficult to prove that the underlying distributions cover the entire normal behaviour, as illustrated by the spurious behaviour learning methods can exhibit where the neural network wrongly makes different classifications of images indistinguishable for humans [30]. A key problem is that classifiers and learning are complex tasks that are hard to prove reliable for humans, in particular through code reviews, which is an often-used procedure during safety certification.

Functional safety standards such as ISO 25119 for agriculture [21] and ISO 13482 for personal care robots [19], are important for the overall functional safety of the robot and for the design of its subsystems. The IEC 61496 [31] standard covers the use of vision in an industrial setting and could thus be used for field robots, as could upcoming standards for performance of vision systems [32], [33]. These standards cover the design of hardware as well as software, which makes it cumbersome to get complex systems and algorithms certified. There are many requirements to a computer vision system: it has to be able to observe a large area; it must be fast, reliable and robust; and it is constrained to function with low computing resources because it normally has to run on embedded hardware. These requirements are on a functional level. Therefore, when aiming for certification, requirements could be inspired by the advances in the automotive domain. As an example, the automotive domain for camera mirrors demands response times of less than 70ms in order to fulfil automotive requirements [34]. As part of the development of the camera mirrors an HRA was performed, where the following hazards were found [34]:

- No image (display completely dark);
- Image does not clearly display scenery according to specification (e.g., adaptation to varying light conditions fails, leading to an overly dark or bright image);
- Frozen image (formerly correct image appears continuously as still image);
- Delayed image w.r.t. reality (more than specification allows);
- Wrong field of view;
- Wrong or unexpected zoom factor (e.g., wide-angle zoom factor for parking mode displayed on highway, making objects appear further away than they actually are);
- Artefacts on display (e.g., double or phantom objects, light spots, dark areas).

Similar challenges are found within robotics, where a multitude of sensors are used like encoders, gyros, sonars and cameras. These sensors can exhibit faults such as sensor bias, locked in place or loss of calibration [35]. This shows that a multitude of issues can occur in the sensor data. Addressing these issues along with developing the solution-oriented algorithm is cumbersome for developers. All these issues are due to the adverse environment where cars and robots have to operate. In the concrete case of mirrors they need to be tested in a multitude of environments [7]. Adapting the environment for off-road test conditions to fit field robots, the hazards become: low sun; uneven surfaces; rain; night drive; and snow.

We use the above challenges from the camera mirror system as inspiration for hazards for robotics domain. The experience from the automotive domain in using functional safety for advanced systems is critical for establishing examples and requirements for the robotic systems. From the automotive domain it is possible to get an overview of the hazards and the test environments where a computer vision system should be validated. In addition to hazard identification and testing in different environments, the computer vision system also has to address performance for detecting obstacles. Obstacles should be understood in a broad definition in which untraversable ground also is categorised as an obstacle (which is even a problem for human operated tractors today [36]), as well as obstacles like humans, animals and trees as examples. Because of safety concerns relating to autonomous robots, it is important to address these issues with compliance to functional safety standards. Standards within software for field robots are used to a very limited extent and are non-existing within computer vision [3]. The lack of standards for field robots is a paradox since autonomous mobile robots rely on robust sensing to react, without robustness the robot may "hallucinate" and respond inappropriately [37]. This issue puts constraints not only on the software but also on the hardware. As an example a RAW image has different degrees of being "RAW" [38]. This difference in RAW can be seen in different A/D converters, gains, and hardware image optimizations. Because of this wide range input, changes in hardware can be problematic.

It is therefore important to comply with functional safety to enable safety verification of the vision pipeline, and to give assurance about the hardware, as well as verifying inputs and outputs for the robotic domain.

Other approaches of designing the software for analysis of perception systems exists [39], but the goal of this high-level system is to tackle the analysis of the image with respect to events, where our approach is based on low-level analysis. The low-level analysis aims to ensure that the data is correct and can be used, and thereby reason about the data. The use of a DSL allows the developer to quickly create code within existing pipelines for verifying the process.

We note that neural networks have received significant attention in the computer vision community, beating other methods in performance on many tasks. Nevertheless, safety certification of neural networks remains an open issue [40]. Artificial Neural Networks (ANNs) should ideally be understandable and readable by humans, while still allowing for individual, meaningful rules [40]. Gupta et al. propose verification and validation of adaptive ANNs [41], although with a focus on control systems. We hypothesise the use of explicit and simple low-level safety rules as a means to be compliant with standards and would enable the safe use of higher-level more complex vision systems for performance.

2.2.2 Software Failure in the Automotive Domain

The automotive domain is focused on functional safety to achieve safety and compliance. However the increased interest in assisted and autonomous functionality for cars exerts pressure on current methods of achieving compliance. Currently the main method for increasing autonomy in cars is to have the driver be the fallback safety in malfunctioning scenarios. This is described as a level 3 autonomy [42]. The current methods for achieving this level is to move the responsibility to the driver, e.g., for Tesla: "*Autosteer is intended for use only on highways and limited-access roads with a fully attentive driver. When using Autosteer, hold the steering wheel and be mindful of road conditions and surrounding traffic. ... Always be prepared to take immediate action. Failure to follow these instructions could cause serious property damage, injury or death*" [43], where the critical point here is the need of a fully attentive driver. This implementation of level 3 autonomy has unfortunately led to a death in a Tesla car crash, which has been investigated by the National Highway Traffic Safety Administration (NHTSA; [43]). The increased autonomy and the environmental perception imposes issues on how safety functions should be implemented. This issue is evident in the crash investigations by NHTSA: "*Both the radar and camera sub-systems are designed for front-to-rear collision prediction mitigation or avoidance. The system requires agreement from both sensor systems to initiate automatic braking.*" [43]. Specifically "*Object classification algorithms in the Tesla and peer vehicles with Autonomous Emergency Braking (AEB) technologies are designed to avoid false positive brake activations. The Florida crash involved a target image (side of a tractor trailer) that would not be a "true" target in the EyeQ3 vision system dataset and the tractor trailer was not moving in the same longitudinal direction as the Tesla, which is the vehicle kinematic scenario the radar system is designed to detect.*" [43]. This example shows the difficulty of environmental sensing and that assumptions made at design time can result in fatalities in production. The statement from the investigation could be interpreted as showing that Tesla has been focussed on availability and not safety, because one knows the uncertainty in the sensors, and for safety the car should have required the user to take over. This however did not happen despite it being required by the system for a full 6 minutes before the crash [44].

The current level 3 autonomy is further problematic because humans will tend to assume that if no faults happen the system is safe and thus reduce awareness [45]. WAYMO is however aiming to field fully autonomous cars corresponding to level 5 [42]. Currently WAYMO are at level 4, which means that the fallback system is a computational system that should be able to bring the car to a safe state, in this case referring to a minimal risk condition that does not require a fully attentive driver as Tesla does. To achieve this WAYMO believes that they have to mix different standards [46], emphasising the issue of achieving compliance for functional safety in autonomous systems. Despite WAYMO having achieved Level 4 autonomy, the operations are strongly restricted to operational design domains, which "*includes geographies, roadway types, speed range, weather, time of day, and state and local traffic laws and regulations*" [46] showing how difficult it is for autonomous machines to deal with dynamic environments.

2.3 Model Driven Engineering

MDE allows for more formalised approaches than standard general purpose programs. Because development is often done across multiple devices, formalisation is an asset for lowering the maintenance cost of developing for many devices, by supporting many different languages or constructs specific to the device. Using MDE for safety ensures that all the different programming languages and devices can be addressed using a single consistent approach. This ensures that rules and code are more reliable across many different implementations. MDE in robotics is an area that currently receives significant attention [47]. Existing research includes control [14], vision [47] and general robot model-driven development [8], [48]. One approach to MDE is to create a DSL [47]. DSLs can be tailored for the specific domain to allow non-software experts to do programming tasks [49]. The narrow scope of the DSL is often claimed as a means of lower learning costs, while still having a high applicability [50]. Although some MDE methods address safety issues [9], in general these issues are not compliant with any safety standards. As an example Bensalem et al. guarantee safety using code generation; the guarantee is mathematical but not in compliance with any standard [13]. This approach is nevertheless one of the few DSLs that explicitly deal with safety. Nordmann et al. found 137 DSLs within robotics, but only 8 within safety and security [51]. Instead focus has been on quality, as with Reichardt et al. who avoid using code generation to improve transparency [6]. The focus on quality and the use of DSLs is however a good way to reduce such issues, because manually written code often is more prone to errors than well-created DSLs [52]. We see code generation as an improvement to reliability, and the possibility of lowering the demands for achieving certification because of mathematical guarantees.

When aiming for safety certification, it is critical that mapping between failures, documentation and code is robust and clear and as a result several attempts exists for extending well-known MDE environments such as RoboML and SysML to incorporate Failure Tree Analysis (FTA) [53], [54] and safety analysis according to IEC 61508 [55]. Other methods like Hazard Operability also exists, which can be seen, e.g., in Zendel et al. [56]. We see FTA as a feasible method, since it is extensively used within more established domains. The FTA can enable the developer to find causality between issues and thereby improve the design of safety functions. Developing the safety functions can however become a large and expensive task. Because the entire hardware and software, that the safety function "touches" has to be included in the certification. Complying with functional safety can be a costly affair both for certification and development, but code generation can generate code compliant to guidelines, e.g., MISRA [57], and improve traceability from textual requirements. Because the DSL can be defined in a more natural language than General Purpose Languages (GPL), code thus improving connection to textual goals. Given the current state of the art, using MDE within functional safety for the computer vision domain is thus a novel idea that has the potential to enable researchers to quickly improve functional safety of robotic systems.

3 Safety Language for Improved Sensor Certification

We believe that the use of explicitly written computer vision rules supports certification authorities in reviews, and thereby provides an increased understanding of the systems safety functions and the functionality. In earlier work we demonstrated that such rules can be used to detect whether the output of a vision pipeline applied to a given image can be trusted [58]. The verified rules were manually implemented, but demonstrate that we can perform the essential task of detecting "bad" images, which is critical since a vision pipeline that is fed bad images may produce arbitrary results and the results produced for such images should therefore not be trusted. The rules for example analyse histogram distributions, high frequency content, or if the images are moving. This paper presents the design and implementation of a DSL for expressing such rules.

3.1 Domain analysis

Following an MDE approach, we adopt the concepts identified in the Robot Perception Specification Language [48]. Here a computer vision pipeline has inputs that are processed by a number of filters to generate outputs. In this setting, the question of whether a computer vision pipeline is performing correctly can be tested in a number of ways. First, the validity of the input image can be tested, if the input data is invalid the result of the vision pipeline cannot be trusted (garbage in, garbage out). Second, the validity of intermediate results can be tested, for a given intermediate result certain outputs are expected at a given stage. Last, the validity of the final result can be tested against other sensors or a ground truth. In this paper we primarily concentrate on testing the validity of the input image, but our approach is general in the sense that intermediate results and the final result can also be tested, as is also demonstrated by our experiments.

The validity of a vision system should not only be considered as a simple boolean property (functioning/failing). For example, the concept of a warning region within safety systems for autonomous robots has been introduced by Mekki-Mokhtar et al. [59], making the system able to categorise three regions "error" (or equivalently: "bad"), "warning", and "good". The split into different regions facilitates later combination of the rules, i.e., if a certain combination of rules produces warnings then the system could also interpret this as an error.

The safety-critical program should target an embedded execution platform running on a high-reliability embedded system, since these often are more suitable for safety certification. The main computer vision pipeline would run on high-performance hardware and communicate selected parts of the input, intermediate and output data to the high-reliability system for validation. The high-reliability system thus only performs the computation needed to validate that the high-performance system is functioning correctly.

3.2 ViSaL concepts

The Vision Safety Language (ViSaL) has been designed to monitor the validity of a computer vision pipeline. The key elements of the ViSaL metamodel are shown in Figure 1, most attributes and details such as expressions are omitted for clarity. A ViSaL program monitors a number of *inputs* each of

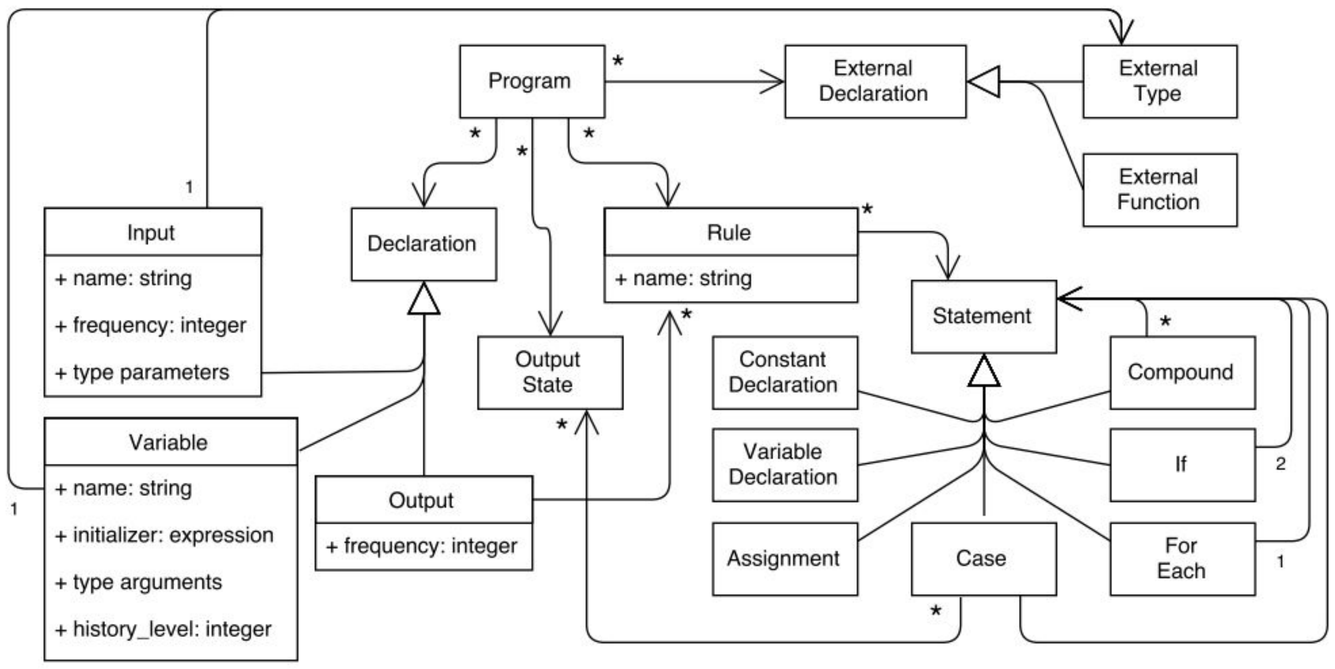

Fig. 1: ViSaL metamodel (UML, selected details omitted, expressions not included)

which correspond to a stream of data being continuously fed into or produced by some aspect of the vision pipeline. The monitoring is performed using *rules* that express if the data is valid at a given point in time according to specific concerns. If the data is invalid a warning or error is signalled for this specific concern.

Output is used to indicate the rules that check the validity of the vision pipeline in terms of different *Output States*. *Variables* record the current state of the vision pipeline as well as a history of these states. The *Rules* are expressed in a minimalistic imperative language and output a classification in terms of output states. The classification result is expressed using *Case* statements, in terms of a numerical quantification of the pipeline data which then can be used for taking decision about the system.

3.3 ViSaL by example

Figure 2 gives an overview of different capabilities in the ViSaL DSL. The example shows a subset of the possible rules in the language, based on the camera mirror hazards, so as to reflect that ViSaL can be used to express functional safety requirements. A ViSaL program starts with initialisation of an input, in this case a `StereoCamera` (line 2), along with an optional timing declaration (end of line 2). The timing declaration creates an implicit rule that the generated code expects input images with a certain frequency, in this case 10Hz. The `StereoCamera` is initialised with the size of the images, again resulting in an implicit rule verifying that the images must be of the correct size (if they are not an error will be signalled at runtime, as with the timing rule). Next the declaration of `states` (line 6) defines output categories for the rules, the ordering indicates which outputs take precedence over others (e.g., if there is just one "bad" result, everything is "bad"). The declaration of `output` (line 8) lists which rules are taken into account, and has a timing parameter which imposes an implicit rule that the `output` has to be produced within the required time. The input from the `StereoCamera` is converted to monochrome images with the `Bayer2Mono` function (line 11). Since the rules rely on a histogram this is also initialised globally making it available for all the rules. Next the individual rules are formulated. Here it is possible to use statements, cases, loops, and call specific functions.

The rules implemented in Figure 2 are designed to demonstrate that the language can perform the essential task of detecting "bad" images. All of the rules use a threshold that defines if the image violates the rules. The thresholds were experimentally determined and their effectiveness demonstrated using a systematic evaluation procedure, where Precision-Recall curves were used to ascertain the applicability for their use in vision systems [58]. Since the thresholds would be considered part of a safety specification of a robot, they are integrated into the program in the case statements.

- The rule "Change Image" (CH) concerns the hazard of a stuck image. CH evaluates if there are any changes in the incoming image compared to the last image. The CH rule randomly selects pixels from the image to do the comparison. The code could evaluate all pixels in an image, but for functional safety certification the hardware

```
//initialising input, which requires size and rate of data.
input StereoCamera(height=752, width=480) camera @ >10Hz;

//Determining the priority of possible "states" for valid outputs for defined rules.
states bad > warn > ok;

//out yields the output from the rules specified,
//which should comply with the specified frequency.
output out = CH, FB, BN, FR, CC @ <10ms;

//Code run every iteration
Image(height=752, width=480) monoLeft = Bayer2Mono (source = camera.left.image);
Histogram h = histogram ( source = monoLeft, bins = 16 );

rule CH { // CHanging image.
 const int count = 5000;
 var int sums = 0;
 //The foreach loop extracts a random set of pixels from an image, and evaluates if the pixels are
 //different compared to a previous existing image.
 foreach Point pos from randomSet(min=0,image=monoLeft, max=monoLeft.pixels.length, n=count):
  if monoLeft.pixels.get(index=pos) != previous(monoLeft.pixels.get(index=pos)):
   sums = sums+1;
 case sums/count {
  [0.9, 1] yield ok;
  [0.6, 0.9[ yield warn;
  else yield bad;
 }
}
rule FB { // Filled Bins ratio of a histogram.
//The field "binSizes" contains a set of the sizes of all bins in the histogram. Set, then extracts
//a subset from "binSizes" and size evaluates the resulting number of bins.
 case size(set(uint x in h.binSizes: x>100)) / size(h) {
  [0.9, 1] yield ok;
  [0.7, 0.9[ yield warn;
  [0, 0.7[ yield bad;
 }
}
rule BN { // BiN distribution, maximum vs. minimum.
 var uint max_min = maximum(h.binSizes) - minimum(h.binSizes);
 case max_min / monoLeft.pixels.length {
  [0, 0.2[ yield ok;
  [0.2, 0.4[ yield warn;
  [0.4, 1] yield bad;
 }
}
rule FR { // Energy Ratio when utilising a filter.
 //The images is filtered using a high-pass butter-worth filter.
 var Freq freq = freqFilter(image=monoLeft, filter=Butter);
 //Compares the frequency before and after the applied filter to how much high-frequency content exists.
 case freq.after / freq.before {
  ]0.0001, 1] yield ok;
  ]0.00005, 0.0001] yield warn;
  [0, 0.00005] yield bad;
 }
}
rule CA { // Component Analysis .
 //Computes the connected components for the light and dark spots in the image.
 var CComp cc_bot = connectedComponents(pixels=monoLeft, exp=lessThan, range=20);
 var CComp cc_top = connectedComponents(pixels=monoLeft, exp=moreThan, range=245);
 //Finds the largest area and divides it with the total pixels.
 case maximum(cc_bot.Largest, cc_top.Largest) / monoLeft.pixels.length {
  ]0.02, 1] yield bad;
  ]0.01, 0.02] yield warn;
  [0, 0.01] yield ok;
 }
}
```

Fig. 2: Implementation of different image analysis rules for verifying the data integrity of images [58].

Declaration	::= Input $\mid$ Variable $\mid$ Type $\mid$ Function $\mid$ Output $\mid$ States
Input I	::= `input` N_T $(\texttt{(}\overline{Argument}\texttt{)})?$ N_I (`@` >? C Unit)?;
Variable V	::= $N_T(\texttt{(}\overline{Argument}\texttt{)})?$ N_V = $\overline{Exp_{\cdot i}}$
External Type T	::= `external type` N_T $(\texttt{(}\ \overline{Argument}\ \texttt{)})?$ $(\{\overline{E}\})?$
External Field E	::= `field` N_E `:` N_T
Function F	::= `external function` N_F `(` $\overline{Parameter}$ `)` `:` N_T
Output O	::= `output` N_{out} = N_R (, N_R)? (`@` >? C Unit)?;
States	::= `states` (N_S >)+ $N_{S_{\mathrm{i}}}$;
Rule R	::= `rule` N_R `{` Statement `}`
Statement	::= Constant $\mid$ LocalDecl $\mid$ ForEach $\mid$ Conditional $\mid$ Case $\mid$ Sequence
ForEach	::= `foreach` N_T N_V `from` E : Statement
Expression Exp	::= C $\mid$ DotExp $\mid$ `previous`(Exp) $\mid$ Exp + Exp $\mid$ ...
DotExp	::= $N_{I\cup L}$ (`.` N_E)*
LocalDecl	::= (`const` $\mid$ `var`) N_T N_{cD} = (Constant $\mid$ E_T $\mid$ V);
Case	::= `case` Exp`{` (Range `yield` N_S ;)+ `}`
Range	::= (`(` $\mid$ `[` $\mid$ `]`) C , C (`)` $\mid$ `[` $\mid$ `]`)
Conditional	::= `if` LogicalExp : Statement
LogicalExp	::= LogicalExp LogicalOp LogicalExp $\mid$ Exp ComparisonOp Exp
ComparisonOp	::= `==` $\mid$ `>=` $\mid$ ...
LogicalOp	::= `&` $\mid$ ...
Sequence	::= Statement*
Unit	::= (`Hz` $\mid$ `ms`)
Argument	::= N_A = Exp
Parameter	::= N_A = N_T
Constant C	$\in \mathbb{R}$
Number N	$\in \mathbb{N}$

Fig. 3: The EBNF grammar of ViSaL. Where the `terminals` and the *non−terminals* of the languages can be seen. $\overline{X}$ corresponds to comma separated list of X. N_X indicates namespace of the corresponding construct.

used is often very limited in performance, which is why we want the rules to be adapted for performance.

- The rule "Filled bins ratio of a histogram" (FB) concerns the hazard of incorrect image exposure. The FB rule is based on histogram analysis. The analysis concerns the relation between the number of bins with pixels divided by the total number of bins.
- The rule "Bin distribution, maximum vs. minimum bin" (BN) concerns the hazard of a covered image (i.e., "lens cap on"). This rule finds the bin with most pixels and subtracts the pixel value from that of the bin with the lowest amount of pixels.
- The rule "Energy ratio before and after high-pass filtering (FR)" concerns the hazard of a blurred image. The rule compares the energy of the image before and after using a filter. This gives a notion about the high-frequency content / sharpness of the image.
- The rule "Component analysis" (CA top & bottom) concerns the hazard of an overexposed image. Connected component analysis is used for finding over exposed (e.g., over exposure) and under exposed(e.g., covered image) spots on the image. The rule is able to catch significant light and dark spots on the image.

These rules exemplify the kinds of properties that can be checked using ViSaL.

3.4 Syntax and Semantics

The EBNF of ViSaL is shown in Figure 3. The syntax largely follows the metamodel (see Figure 1). The history level of variables is not declared explicitly, but is determined by the compiler by analysing the use of the "previous" keyword. External types and functions declared in library, interface to external C++ code. Ranges have a flexible syntax supporting different mathematical notations for specifying intervals.

The semantics of ViSaL are represented by the algorithm shown in Figure 4. A ViSaL program first initialize `inputs I`, `states OS` and `output O`, where the input and output have optional frequency parameter to impose run-time constraints. The `inputs I` are then analysed with respect to correct data, in terms of size, as well as timing if frequency has been specified. Then each `rule R` is iterated over, first with regards to the rules specific code `R.statements`, and

```
initialize inputs I @ freq F
initialize output O @ freq F

repeat
  foreach input i in I
    assert correctFormat(i.data)
    assert (i.curTime - i.lastTime) < i.F
    i.lastTime = i.curTime

  foreach rule r in R:
    evaluate r.statements
    O[r] = r.caseResult

    assert (O.curTime - O.lastTime) < O.F
    O.lastTime = O.curTime
```

Fig. 4: The semantics of a ViSaL program.

then with respect to the case selection `R.caseResult`. If the output of the `R.caseResult` does not fall within expected thresholds, then an error case will be added instead. Finally the optional frequency parameter for the output is tested. The output is finally made available to the enclosing system. The decision system can then decide to trust or not trust the data flow, or in worst case demand a return to a safe state, e.g., stopping the system.

The time specification for inputs has an undefined semantics if history is used on different inputs with different frequencies. Therefore ViSaL does not allow to mix inputs with different frequencies in a program if history is utilized in a rule, since the input with lower frequency in a history rule would evaluate the same image twice, which could introduce spurious errors.

3.5 Implementation

A single named entrypoint is generated from ViSaL, which is a main function that first initialises all variables, their history, and a timer (but only to the extent that these are used by the DSL program). Then follows a main loop that continuously checks time and image size constraints (if appropriate) and then evaluates each rule in turn, while collecting information about any errors triggered. In the end of the loop the variable history is updated, as required for the DSL program. When experimenting off-line the results of all the rules are stored in a text file, ensuring new files for each run.

In previous work, the rules implemented by the ViSaL program shown in Figure 2 were informally described and then subsequently implemented manually in MATLAB for quantitative evaluation [58]. We can now automatically generate the C++ code with the same functionality, allowing the developer to specify rules in high-level ViSaL specifications and then generate C++ with the same functionality as the Matlab code.

ViSaL has been implemented using the xtext language workbench [60]. The current implementation generates C++ code compatible with the C++11 standard using OpenCV 3.1.0 [61]. While this is appropriate for the hardware currently targeted, a future implementation could also generate more basic C code using a dedicated image processing library, which would be more applicable for safety certification. Note that not all of OpenCV is available to the developer, but rather a subset of the library is utilized by ViSaL, expressed using ViSaL external types. For safety ViSaL does not have a generic interface for the OpenCV library, but the compiler uses a modular design that makes it easy to support new datatypes or operators based on OpenCV. Allowing unrestricted access to OpenCV would make it very difficult to argue for safety during certification.

4 Evaluation

In this section we evaluate ViSaL with respect to performance and qualitative issues. Section 4.1 compares manually written code to generated code from ViSaL. In terms of performance on an embedded hardware platform. Section 4.2 examines claimed ViSaL benefits with respect to understandability.

Because ViSaL is aimed at safety-critical hardware it is important to evaluate the code on embedded hardware. We use an NVidia Jetson board [62], which is in a family with the automotive grade hardware variant. While the TX1 board is not safety compliant, the automotive variations of the board denoted PX [63], are certifiable. The reason why the PX board is able to be safety certified is that it incorporates certified processing units for providing the guarantees needed. We test on a TX1 development board, as it resembles the performance available on a PX board and make benchmarks more realistic, while we are aware that the PX board will have less computational resources on the safety-critical unit.

4.1 ViSaL Evaluation

We made recordings with the NVidia Jetson TX1 board connected to a CLAAS stereo camera [64]. The recordings were made using a setup based on ROS [65]. The images are recorded in scenarios that correspond to a selection of the issues stated by Schmidt et al. with regards to automotive camera mirror systems for cars [34], since similar hazards often can be seen in general perception systems for robots.

The benchmark evaluation is only done on one CPU core and not utilising the GPU. The benchmark process is as follows:

- Clean restarted board, with ROS Indigo [66] installed with Ubuntu 14.04 as OS.
- The benchmark is run for both code bases, where the initial execution in both cases is discarded as warm-up.
- The benchmark consists of evaluating 99 images, a sample of which is presented in Figure 5, and is executed 50 times for each program, excluding warm-up.

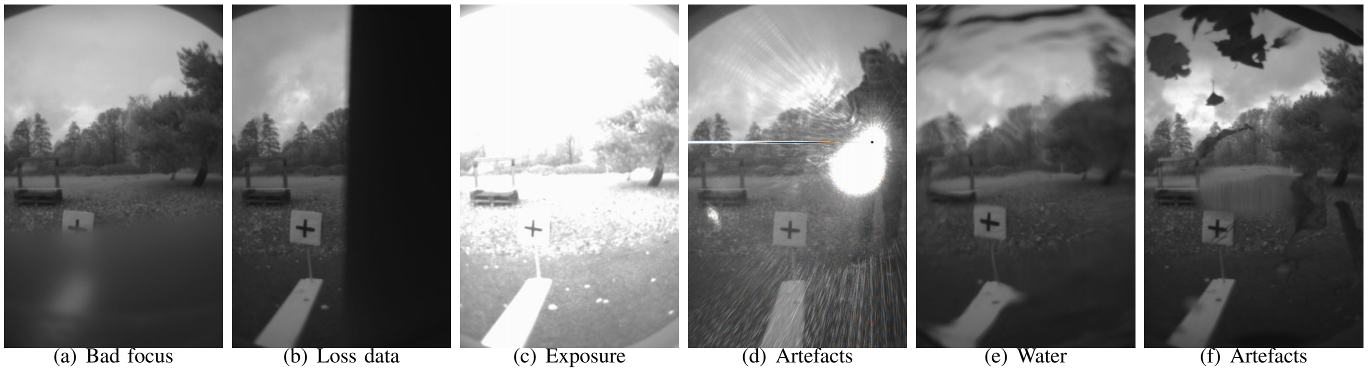

Fig. 5: Example images representative of issues found when using the experimental stereo camera setup.

- The output from both processes are compared with a checksum to assert that we got the same results for all the runs.

We then extract the mean and standard deviation of the timings in both cases referring to Table 1. There is an indication that the generated code performs a bit worse than the manual code. We investigate this indication by using paired-samples t-test [67]. The paired-samples allows for testing with small sample sizes compared to two-sample t-test, while also reducing the signal-to-noise ratio that determines statistical significance. We investigate H_0 if the two sets have equal means, comparing against $H_{1,1}$ alternative hypothesis that the population mean of ViSaL is larger than for the manual code. We found as was evident from the Table 1, that the mean runtime for ViSaL-generated code is higher than the manual code (p-value = $8e^{-14}$). This could be a result of an extra for-loop in the ViSaL code for the rule FR, however the variance is still lower which is positive. The quantitative results thus show that there is a limited cost to using the generated code. This is significant because the generated code can be viewed as being as good as manually written code, making it a feasible replacement.

Looking at the percentage impact of the individual rules (see Table 2). FB, creates the initial histogram of the images, which is why it so much slower than BN. R consists of transforming the image and filtering it, which is computationally heavy on a CPU. The performance would probably increase if the GPU or a dedicated processing unit for vision was utilized. A GPU is however not present in the safety-critical processor, therefore a platform-specific dedicated processing units for vision would be the preferred method for improving the performance. We expect that ViSaL would be suitable for generating code for such a platform, significantly reducing development time compared to C++.

4.2 Qualitative Evaluation

The results from the quantitative tests of the generated code show performance being on-par with the manually written code. We now investigate the ViSaL qualitatively to understand if there is any added benefit of using ViSaL. The ViSaL program that is tested takes up 52 lines of code, excluding a few generic library declarations not shown in Figure 2. The resulting generated C++ code is 186 lines. The ViSaL program was originally implemented in C++, with the same functionality as provided by this ViSaL program, but required 147 lines of code to implement. Using ViSaL developers are thus able to write the same functionality with fewer lines of code compared to C++. We expect that the ViSaL code is as easy to read as the equivalent C++ code, and that a wider range of properties can be guaranteed statically for ViSaL than for C++, which we have verified based on a systematic study [68]. Based on our systematic study [68] we extended the language with more rules, optimised range notation, history, and most importantly the yield constructs, shown in Figure 2. We found that the history construct and

TABLE 1: Overview of the mean, μ, and standard deviation, σ, runtimes for the manual written and auto generated code. The results in the table are in milliseconds.

	Mean μ	Standard deviation σ
Manual	348.3708	4.4377
ViSaL	349.5751	2.8127

TABLE 2: Overview of the rules percentage of an execution of the results of ViSaL and manual in Table 1, the values are rounded.

	ViSaL	Manual
FB	16.16	15.72
CA	0.81	0.88
CH	0.010	0.017
BN	0.002	0.001
FR	83.02	83.46

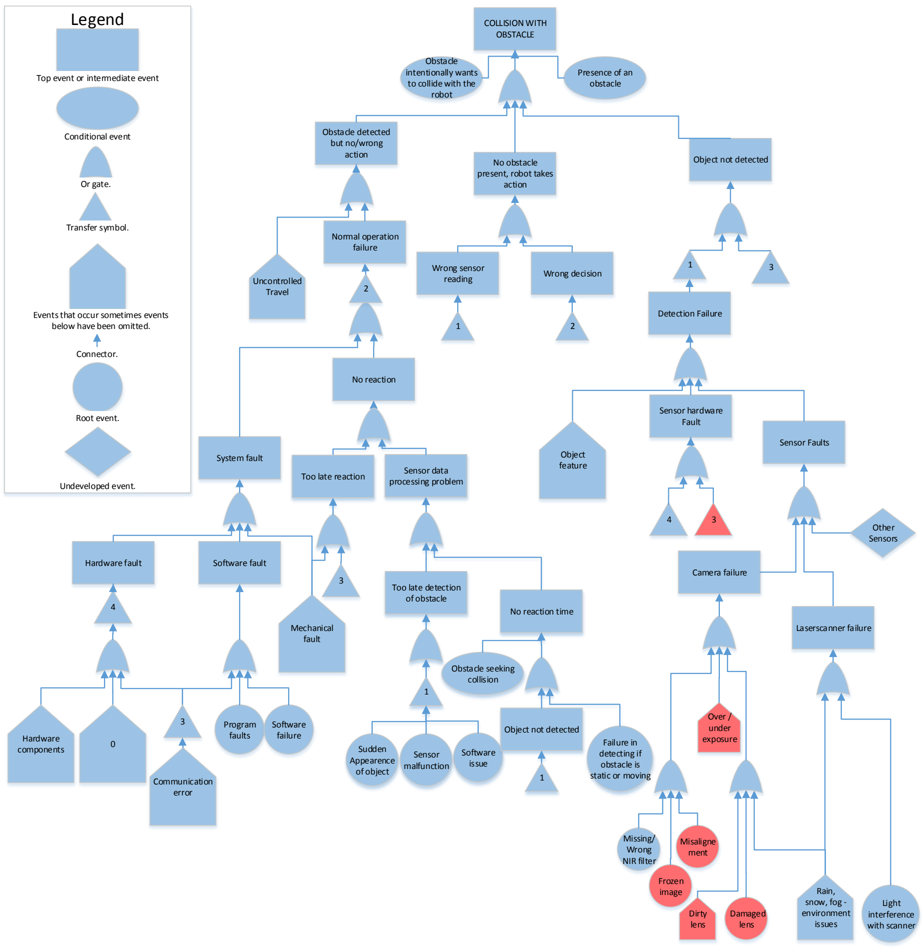

Fig. 6: A FTA of possible issues leading to the malfunction of colliding with an obstacle.

the case structure removed errors that were difficult to uncover for participants. Lastly we found that parameter naming in some instances could lower readability as the variable was disregarded, prompting a change to this part of the language as a result. The ViSaL program shown in Figure 2 uses this new and improved syntax.

In order to demonstrate that the rules that can be expressed by ViSaL program can perform many kinds of classifications that would be useful for maintaining the safety of a mobile robot, we have implemented an additional set of rules that are significantly different from previously introduced rules:

1) **3D Object (3D):** We insert a physical marker at a fixed location in the scene, enabling the rule to verify that the 3D points from the stereo matching are usable. Because it is known that a specific number of points should exist at the physical marker location. Example hazard: Lost stereo calibration.
2) **Optical Flow (OF):** Optical flow using Lucas Kanade [69] uncovers if an image is changing. This is done by evaluating how many areas of the image are moving above a certain threshold. Example hazard: Stuck image.
3) **Entropy (EN):** Shannon entropy is a measure of the unpredictability of the image. This measure is added as a function, but also possible to implement through ViSaL expressions. The concept is to evaluate how much information exists in the current image and compare it to the previous image to understand if the information level has changed drastically. The functionality was based on input from a systematic study [68]. Example hazard: Images which are noise.

The rules were tested and the generating code from ViSaL was used with the dataset seen in this paper to verify that they are able to detect issues.

The output of the different rules will enable the system to analyse the trustworthiness of the sensor and act accordingly. Similar concepts could be done for a lidar, as shown by an initial concept presented in [70], this initial DSL however only focused on cameras.

5 Discussion

The language that we propose reduces the lines of code to 52 lines, from 186 lines in manually written C++ code, for the same functionality. This can increase the ease of development, but the most important advantage is the easy compliance with software guidelines. The generated code could be made to comply with MISRA [57] rules to extend the trustworthiness of the code generator, and thereby pave the way for compliance with standards. MISRA is a guideline of which sub-parts of the C or C++ language (depended on the MISRA standard) should and should not be used, along with the requirement of documentation, i.e., how the process has been followed, all of which could be generated.

At present time the generated code is however not certifiable, further steps are needed, in particular a guarantee that the generated code is the same as the ViSaL code. In workshops with TÜV we learned this could be accomplished by a separate compiler that can map C++ to ViSaL, where the two compilers are not allowed to share code base [18]. Therefore the current DSL should be viewed as a concept to achieve certification, however it is not limited to scenarios like the camera mirror case, but can also be used for more advanced perception pipelines. This can be done by defining and using low-level rules according to system specific HRAs and FTAs, and thereby serve to ensure that high-level and advanced perception systems perform as expected and are trustworthy. To show the possible explicit mapping we have made an FTA of common issues in perception systems. The FTA is created with respect to the malfunction of a field robot hitting an obstacle, which is shown in Figure 6. The symbols in the FTA follows an fault tree handbook for aerospace applications [71]. A specific symbol is the transfer symbol means that if there is information prior to the symbol, then it describes the transfer symbol, if there is no description it is a copy-paste of an earlier description. The FTA is read from the bottom up, meaning that the top event “collision with an obstacle” is a result of faults propagating up through the tree.

An FTA creates a visual record of a system, that illustrates the relations between issues that can lead to a specific failure. The FTA in Figure 6 is an extract from documentation made for a field robot. The extract covers a specific failure: colliding with an obstacle, which is a result from a safety goal. The FTA omits information that is not relevant for ViSaL, specifically issues with communication, hardware (e.g., memory error/-corruption), and mechanical issues. The bottom-right corner of the FTA explicitly states low-level issues for images; exposure, frozen image, and misalignment. These issues are with respect to the FTA, which in worst case can lead to the overall catastrophic failure of colliding with and obstacle. The low-level issues reference are root causes in the FTA, and need to be addressed for certification. ViSaL enables the developer to create explicit rules that addresses these specific issues.

Comparing the FTA in Figure 6 with the ViSaL example code in Figure 2, certain issues can be connected to the implemented rules, marked in a different colour. The frozen image failure, can be directly mapped to rule CH (line 14-27). The other rules implemented in the example addresses over and under exposure, which is also an issue in the FTA. In addition the timings specified on input and output (line 2 and 6 respectively) in ViSaL, can detect issues with communication between sensor and processing unit, since it expects a message with a certain frequency. Thereby ensuring if there is “no” or a “late response” from the sensor, that the decision system receives the information from the ViSaL process, and allows the decision system to act accordingly. The reaction could be to rely on other sensors or reach a safe state, e.g., stopping the system. The ability to easily match implemented functionality and requirements, be that safety-critical or not, is an important aspect of ViSaL. This property has been investigated in a systematic study [68], where we found that ViSaL performed better than C++for textual matchings.

6 Future Work

Future work concerns the experimental validation of ViSaL by using an automatically generated safety monitor with a vision pipeline installed on a tractor or field robot working in a real-life scenario. We will also experimentally verify if combination of rules can increase detection of errors in ViSaL and allow for it to be expressed. Future work for the language includes further static guarantees, e.g., regarding data type inference or worst-case execution time. could be to infer types and verify code using inference. ViSaL is used to express simple rules where thresholds for the vision must be specified. The Matlab code specific for the rules could also be generated to improve traceability.

Currently we utilise OpenCV as an underlying framework, but other frameworks could be supported in the future by using different target platforms for compilation. We consider NVidia Drive PX and MobileEye SoCs as interesting target platforms, but the question of which vision framework to target on these platforms is future work.

ViSaL could therefore be used to guarantee sensor inputs and be a facilitator for safe navigation. To enable safe navigation additional algorithms have to be included, e.g., utilising a perimeter calculation based on Täubig et al.’s concept [72]. The speed and detection zones could further be extended by the notion of *passive motion safety* [73] in connection with guaranteeing a safety zone [72]. Introducing these capabilities when verifying a perception systems for navigation could improve certification. An extension to ViSaL to test hardware issues, e.g., failure in calculations, could be to use stored images and their results, to see if the output matches earlier recorded results, to verify the process during operation. Further saving the known image different places in memory could also allow for test of the memory system, thus improving the certification flow, because more aspects of issues in the FTA is covered.

The concepts could be transferred to other sensors or perception pipelines. We have conducted experiments with lidars [70], where the number of working laser channels was assessed by using a fixed landmark, similar to the idea of the 3D rule. Furthermore, many perception pipelines are reliant on neural networks where semantic rules could make sense, i.e., sky should be above the ground or a certain percentage of ground should be visible in the image. Both cameras and lidars have however been seen to be susceptible to attacks [74]. The Proposed counter measures can currently not be implemented by ViSaL, the reason is two-fold, first that the language does not support features such as randomising ping waveform; and some sensors, e.g., Velodyne and IBEO, emit pre-processed signals and augmenting the emitted signals is not possible at present time. Nevertheless this will not thwart all possible spoofing attempts [74]. We believe however that the issue of attacking cameras with over-exposed light sources, which cameras has been shown to be vulnerable to [75], [76], will be caught by ViSaL, e.g., using rules such as 3D (known object) or CA (connected component). The issue of attacks is not limited to perception sensors; gyroscopes have also been proven to be susceptible [77] as have encoders [78] Investigations into perception sensors safety and security is rare [74], and therefore more research is needed within safety, security and the cross-domain of the two. It is interesting to investigate how small an area should be detected, following that more data is needed for testing the different use-cases, i.e., normal-, faulty- and under-attack operation.

7 Conclusion

The use of vision-based sensing is critical for many uses of mobile robots in dynamic environments, and yet the issue of systematically designing, implementing and certifying safety of vision systems remains largely unaddressed. Based on the concept of simple and explicit rules for detecting failures of a computer vision pipeline, we have designed the ViSaL DSL for enabling concise and efficient implementations of such rules, as presented in this paper. The use of a DSL enables automatic code generation and is critical for automating other aspects of the process, such as determining the required error/warning thresholds and automatically integrating with other model-driven approaches to programming of vision pipelines.

References

[1] Patrick J. Griffin, "Safety and Health in Agriculture "Farming - a hazardous occupation how to improve health & safety?"," 2013. [Online]. Available: http://www.europarl.europa.eu/document/activities/cont/201303/20130321ATT63633/20130321ATT63633EN.pdf 1

[2] S.-Y. Yang, S.-M. Jin, and S.-K. Kwon, "Remote control system of industrial field robot," in *Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on*. IEEE, 2008, pp. 442–447. 1

[3] J. T. M. Ingibergsson, U. P. Schultz, and M. Kuhrmann, "On the use of safety certification practices in autonomous field robot software development: A systematic mapping study," in *Product-Focused Software Process Improvement*. Springer, 2015, pp. 335–352. [Online]. Available: http://dl.acm.org/citation.cfm?id=2972500 1, 2.1, 2.2.1

[4] U. Frese and H. Hirschmüller, "Special issue on robot vision: what is robot vision?" *Journal of Real-Time Image Processing*, vol. 10, no. 4, pp. 597–598, 2015. [Online]. Available: http://www.irisa.fr/lagadic/pdf/2015_ijrr_editorial.pdf 1

[5] J. Carlson, R. R. Murphy, and A. Nelson, "Follow-up analysis of mobile robot failures," in *International Conference on Robotics and Automation (ICRA)*, vol. 5. IEEE, 2004, pp. 4987–4994. [Online]. Available: http://ieeexplore.ieee.org/iel5/9126/28923/01302508.pdf 1

[6] M. Reichardt, T. Föhst, and K. Berns, "On software quality-motivated design of a real-time framework for complex robot control systems," in *International Workshop on Software Quality and Maintainability*, 2013. [Online]. Available: http://rrlab.cs.uni-kl.de/fileadmin/Literatur/Reichardt13.pdf 1, 2.3

[7] D. Schmidt, "Guest Editor's Introduction: Model-Driven Engineering," *Computer*, vol. 39, no. 2, pp. 25–31, Feb. 2006. [Online]. Available: http://ieeexplore.ieee.org/document/1597083/ 1, 2.2.1

[8] A. Steck, A. Lotz, and C. Schlegel, "Model-driven engineering and run-time model-usage in service robotics," in *ACM SIGPLAN Notices*, vol. 47. ACM, 2011, pp. 73–82. [Online]. Available: http://dl.acm.org/citation.cfm?id=2047875 1, 2.3

[9] M. Adam, M. Larsen, K. Jensen, and U. Schultz, "Rule-based dynamic safety monitoring for mobile robots," *Journal of Software Engineering for Robotics*, vol. 7, no. 1, pp. 121–141, 7 2016. [Online]. Available: https://joser.unibg.it/ 1, 2.3

[10] A. Santosuosso, C. Boscarato, F. Caroleo, R. Labruto, and C. Leroux, "Robots, market and civil liability: A european perspective," in *RO-MAN*. IEEE, 2012, pp. 1051–1058. [Online]. Available: http://ieeexplore.ieee.org/document/6343888/ 1, 2.1

[11] TC 23, "Standards," International Organization for Standardization, International Standard, 2015. [Online]. Available: http://www.iso.org/iso/home/standards.htm 1

[12] A. De Cabrol, T. Garcia, P. Bonnin, and M. Chetto, "A concept of dynamically reconfigurable real-time vision system for autonomous mobile robotics," *International Journal of Automation and Computing*, vol. 5, no. 2, pp. 174–184, 2008. [Online]. Available: http://link.springer.com/article/10.1007/s11633-008-0174-0 1

[13] S. Bensalem, L. da Silva, M. Gallien, F. Ingrand, and R. Yan, "Verifiable and correct-by-construction controller for robots in human environments," in *seventh IARP workshop on technical challenges for dependable robots in human environments (DRHE)*, 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=2702098 1, 2.3

[14] S. Adam, M. Larsen, K. Jensen, and U. P. Schultz, "Towards Rule-Based Dynamic Safety Monitoring for Mobile Robots," in *Simulation, Modeling, and Programming for Autonomous Robots*, ser. Lecture Notes in Computer Science. Springer, 2014, no. 8810, pp. 207–218. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-319-11900-7_18 1, 2.3

[15] J. T. M. Ingibergsson, D. Kraft, and U. P. Schultz, "Safety computer vision rules for improved sensor certification," in *2017 First IEEE International Conference on Robotic Computing (IRC)*, April 2017, pp. 89–92. [Online]. Available: http://ieeexplore.ieee.org/document/7926520/ 1

[16] ISO/COPOLCO, "Safety aspects – guidelines for their inclusion in standards," International Organization for Standardization, International Standard ISO 51:2014, 2014. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso-iec:guide:51:ed-3:v1:en 2.1

[17] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE transactions on dependable and secure computing*, vol. 1, no. 1, pp. 11–33, 2004. 2.1

[18] D. Zenke, D. J. Listner, and J. T. M. Ingibergsson, "Meeting on safety in sensor systems with employees from TÜV NORD," Personal communication, 2016. 2.1, 5

[19] TC 184, "Robots and robotic devices - Safety requirements for personal care robots," International Organization for Standardization, International Standard ISO 13482:2014, 2014. [Online]. Available: https://www.iso.org/standard/53820.html 2.1, 2.2.1

[20] S. Dogramadzi, M. E. Giannaccini, C. Harper, M. Sobhani, R. Woodman, and J. Choung, "Environmental hazard analysis - a variant of preliminary hazard analysis for autonomous mobile robots," *Journal of Intelligent &*

Robotic Systems, vol. 76, no. 1, pp. 73–117, 2014. [Online]. Available: http://link.springer.com/article/10.1007/s10846-013-0020-7 2.1

[21] TC 23, "Tractors and machinery for agriculture and forestry – safety-related parts of control systems," International Organization for Standardization, International Standard ISO 25119-2010, 2010. [Online]. Available: https://www.iso.org/standard/45050.html 2.1, 2.2.1

[22] TC 22, "Road Vehicles Functional Safety," International Organization for Standardization, International Standard ISO 26262:2011, 2011. [Online]. Available: http://www.iso.org/iso/catalogue_detail?csnumber=43464 2.1

[23] K. Hedenberg and B. Åstrand, "Safety standard for mobile robots-a proposal for 3d sensors." in *ECMR*, 2011, pp. 245–252. [Online]. Available: https://www.researchgate.net/publication/266348494_Safety_standard_for_mobile_robots_-a_proposal_for_3D_sensors 2.1

[24] EN 1525, *Safety of industrial trucks. Driverless trucks and their systems*. DIN, 1998. [Online]. Available: http://shop.bsigroup.com/ProductDetail/?pid=000000000001609961 2.1

[25] C. Fields, R. David, and P. Nielsen, "Defense science board 2016 summer study on autonomy," *Defense Science Board*, 2016. [Online]. Available: https://www.hsdl.org/?view&did=794641 2.1

[26] K. Heckemann, M. Gesell, T. Pfister, K. Berns, K. Schneider, and M. Trapp, "Safe automotive software," in *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*. Springer, 2011, pp. 167–176. [Online]. Available: http://www.springerlink.com/index/X635334588485425.pdf 2.2

[27] F. Stein, "The challenge of putting vision algorithms into a car," in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, June 2012, pp. 89–94. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/6238900/ 2.2

[28] R. Wang and B. Bhanu, "Learning models for predicting recognition performance," in *Tenth IEEE International Conference on Computer Vision (ICCV)*, vol. 2. IEEE, 2005, pp. 1613–1618. [Online]. Available: http://www.comp.leeds.ac.uk/me/Publications/bmvc09.pdf 2.2.1

[29] P. Zhang, J. Wang, A. Farhadi, M. Hebert, and D. Parikh, "Predicting failures of vision systems," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3566–3573. [Online]. Available: http://ieeexplore.ieee.org/document/6909851/ 2.2.1

[30] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015, pp. 427–436. [Online]. Available: https://arxiv.org/abs/1412.1897 2.2.1

[31] TC 44, "Safety of machinery – electro-sensitive protective equipment," International Electronical Commission, International Standard IEC 61496-2012, 2012. [Online]. Available: https://webstore.iec.ch/publication/5500 2.2.1

[32] TC 23, "Agricultural machinery and tractors – Safety of highly automated machinery," International Organization for Standardization, International Standard ISO/DIS 18497, 2014. [Online]. Available: https://www.iso.org/standard/62659.html 2.2.1

[33] TC 127, "Earth-moving machinery – autonomous machine system safety," International Organization for Standardization, International Standard ISO 17757-2015, 2015. [Online]. Available: https://www.iso.org/standard/60473.html 2.2.1

[34] E. A. Schmidt, H. Hoffmann, R. Krautscheid, M. Bierbach, A. Frey, J. Gail, and C. Lotz-Keens, "Camera-monitor systems as a replacement for exterior mirrors in cars and trucks," in *Handbook of Camera Monitor Systems*. Springer, 2016, pp. 369–435. [Online]. Available: http://link.springer.com/chapter/10.1007_2F978-3-319-29611-1_12 2.2.1, 4.1

[35] M. J. Daigle, X. D. Koutsoukos, and G. Biswas, "Distributed diagnosis in formations of mobile robots," *IEEE Transactions on Robotics*, vol. 23, no. 2, pp. 353–369, 2007. [Online]. Available: http://ieeexplore.ieee.org/document/4154826/ 2.2.1

[36] Agricultural Statistics Board, "Agricultural Safety: 2009 Injuries to Adults on Farms," 2013. [Online]. Available: http://usda.mannlib.cornell.edu/usda/current/AduInj/AduInj-05-30-2013.pdf 2.2.1

[37] R. R. Murphy and D. Hershberger, "Handling sensing failures in autonomous mobile robots," *The International Journal of Robotics Research*, vol. 18, no. 4, pp. 382–400, 1999. [Online]. Available: http://ijr.sagepub.com/content/18/4/382.short 2.2.1

[38] M. S. Brown and S. J. Kim, "Understanding the In-Camera Image Processing Pipeline for Computer Vision," 2015. [Online]. Available: http://www.comp.nus.edu.sg/~brown/ECCV_Tutorial_Color.pdf?bcsi_scan_71efa5f77394d2ae=0&bcsi_scan_filename=ECCV_Tutorial_Color.pdf 2.2.1

[39] F. Heintz, J. Kvarnström, and P. Doherty, "Bridging the sense-reasoning gap: Dyknow–stream-based middleware for knowledge processing," *Advanced Engineering Informatics*, vol. 24, no. 1, pp. 14–26, 2010. [Online]. Available: http://ceur-ws.org/Vol-466/sr2009_submission_1.pdf 2.2.1

[40] Z. Kurd and T. Kelly, "Establishing safety criteria for artificial neural networks," in *Knowledge-Based Intelligent Information and Engineering Systems*. Springer, 2003, pp. 163–169. [Online]. Available: https://www-users.cs.york.ac.uk/tpk/kes2003.pdf 2.2.1

[41] P. Gupta, K. Loparo, D. Mackall, J. Schumann, and F. Soares, "Verification and validation methodology of real-time adaptive neural networks for aerospace applications," in *International Conference on Computational Intelligence for Modeling, Control, and Automation*, 2004. [Online]. Available: https://ti.arc.nasa.gov/m/profile/schumann/PDF/GLSS2004.pdf 2.2.1

[42] SAE, "Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems - j3016," http://standards.sae.org/j3016_201401/, 2016. 2.2.2

[43] J. Q. Kareem Habib and S. Ridella, "Odi resume," 2016. [Online]. Available: https://static.nhtsa.gov/odi/inv/2016/INCLA-PE16007-7876.PDF 2.2.2

[44] D. P. Y. Joseph A. Gregor and M. Schwall, "Driver assistance system," 2017. [Online]. Available: https://dms.ntsb.gov/public/59500-59999/59989/604889.pdf 2.2.2

[45] S. Haddadin, A. Albu-Schaffer, A. De Luca, and G. Hirzinger, "Collision detection and reaction: A contribution to safe physical human-robot interaction," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 3356–3363. 2.2.2

[46] WAYMO, "ON THE ROAD TO FULLY SELF-DRIVING Waymo Safety Report," October 2017. [Online]. Available: https://waymo.com/safetyreport/ 2.2.2

[47] N. Hochgeschwender, S. Schneider, H. Voos, and G. K. Kraetzschmar, "Declarative Specification of Robot Perception Architectures," in *Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2014, pp. 291–302. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-319-11900-7_25 2.3

[48] N. Hochgeschwender, L. Gherardi, A. Shakhirmardanov, G. K. Kraetzschmar, D. Brugali, and H. Bruyninckx, "A model-based approach to software deployment in robotics," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 3907–3914. [Online]. Available: http://ieeexplore.ieee.org/document/6696915/ 2.3, 3.1

[49] D. Ghosh, *DSLs in action*. Manning Publications Co., 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1965333 2.3

[50] G. Fischer, E. Giaccardi, Y. Ye, A. G. Sutcliffe, and N. Mehandjiev, "Meta-design: a manifesto for end-user development," *Communications of the ACM*, vol. 47, no. 9, pp. 33–37, 2004. 2.3

[51] A. Nordmann, N. Hochgeschwender, D. L. Wigand, and S. Wrede, "A survey on domain-specific modeling and languages in robotics," *Journal of Software Engineering in Robotics*, vol. 7, no. 1, 2016. [Online]. Available: https://joser.unibg.it/ 2.3

[52] E. D. Berger, "Software needs seatbelts and airbags," *Communications of the ACM*, vol. 55, no. 9, pp. 48–53, 2012. [Online]. Available: http://queue.acm.org/detail.cfm?id=2333133 2.3

[53] K. Khodabandehloo, "Analyses of robot systems using fault and event trees: case studies," *Reliability Engineering & System Safety*, vol. 53, no. 3, pp. 247–264, 1996. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S095183209600052X 2.3

[54] S. Lee and Y. Yamada, *Risk Assessment and Functional Safety Analysis to Design Safety Function of a Human-Cooperative Robot*. INTECH Open Access Publisher, 2012. [Online]. Available: https://pdfs.semanticscholar.org/ff87/af8d4134b71be29ea4a18df6fce7f3ad69fa.pdf 2.3

[55] N. Yakymets, S. Dhouib, H. Jaber, and A. Lanusse, "Model-driven safety assessment of robotic systems," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 1137–1142. [Online]. Available: http://ieeexplore.ieee.org/iel7/6679723/6696319/06696493.pdf 2.3

[56] O. Zendel, M. Murschitz, M. Humenberger, and W. Herzner, "Cv-hazop: Introducing test data validation for computer vision," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2066–2074. 2.3

[57] MISRA, *MISRA-C Guidelines for the Use of the C Language in Critical Systems*. Motor Industry Software Reliability Association, 2012. [Online]. Available: http://www.misra.org.uk/Publications/tabid/57/Default.aspx 2.3, 5

[58] J. T. M. Ingibergsson, D. Kraft, and U. P. Schultz, "Explicit image quality detection rules for functional safety in computer vision," in *12th International Conference on Computer Vision Theroy and Applications (VISAPP)*, Marts 2017, p. 12. [Online]. Available: http://www.scitepress.org/DigitalLibrary/PublicationsDetail.aspx?ID=SzBMbPkrZL0=&t=1 3, 3.3, 2, 3.5

[59] A. Mekki-Mokhtar, J.-P. Blanquart, J. Guiochet, D. Powell, and M. Roy, "Safety trigger conditions for critical autonomous systems," in *18th Pacific Rim International Symposium on Dependable Computing*. IEEE, 2012, pp. 61–69. [Online]. Available: http://ieeexplore.ieee.org/document/6385071/ 3.1

[60] M. Eysholdt and H. Behrens, "Xtext: implement your language faster than the quick and dirty way," in *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*. ACM, 2010, pp. 307–309. [Online]. Available: http://dl.acm.org/citation.cfm?id=1869625 3.5

[61] Itseez, "Open source computer vision library," https://github.com/itseez/opencv, 2015. 3.5

[62] NVIDIA, "Embedded systems," 2016. [Online]. Available: http://www.nvidia.com/object/jetson-tx1-module.html 4

[63] ——, "Embedded systems," 2017. [Online]. Available: http://www.nvidia.com/object/drive-px.html 4

[64] CLAAS, "Culti Cam Picture," 2015. [Online]. Available: http://www.claas-e-systems.com/en/oem-products/camera/ 4.1

[65] 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. [Online]. Available: http://www.willowgarage.com/sites/default/files/icraoss09-ROS.pdf 4.1

[66] ——, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009. [Online]. Available: http://www.willowgarage.com/sites/default/files/icraoss09-ROS.pdf 4.1

[67] D. W. Zimmerman, "Teacher's corner: A note on interpretation of the paired-samples t test," *Journal of Educational and Behavioral Statistics*, vol. 22, no. 3, pp. 349–360, 1997. [Online]. Available: http://journals.sagepub.com/doi/abs/10.3102/10769986022003349 4.1

[68] J. T. M. Ingibergsson, J. Sunshine, and U. P. Schultz, "Readability study of a domain specific language: Process and outcome," in *Submitted for publication*, 2017. [Online]. Available: https://www.dropbox.com/s/e9j70dgprb2r0f5/ReadabilityStudy_Anon.pdf?dl=0 4.2, 3, 5

[69] B. D. Lucas, T. Kanade *et al.*, "An iterative image registration technique with an application to stereo vision." in *IJCAI*, vol. 81, 1981, pp. 674–679. 2

[70] J. T. I. Mogensen, S.-D. Suvei, M. K. Hansen, M. P. Christiansen, and U. P. Schultz, "Towards a dsl for perception-based safety systems," in *6th International Workshop on Domain-Specific Languages and models for Robotic systems*, 2015. [Online]. Available: https://www.researchgate.net/publication/283457486_Towards_a_DSL_for_Perception-Based_Safety_Systems 4.2, 6

[71] M. Stamatelatos, W. Vesely, J. Dugan, J. Fragola, J. Minarick, and J. Railsback, "Fault tree handbook with aerospace applications," 2002. 5

[72] H. Täubig, U. Frese, C. Hertzberg, C. Lüth, S. Mohr, E. Vorobev, and D. Walter, "Guaranteeing functional safety: design for provability and computer-aided verification," *Autonomous Robots*, vol. 32, no. 3, pp. 303–331, Apr. 2012. [Online]. Available: http://link.springer.com/10.1007/s10514-011-9271-y 6

[73] S. Bouraine, T. Fraichard, and H. Salhi, "Provably safe navigation for mobile robots with limited field-of-views in dynamic environments," *Autonomous Robots*, vol. 32, no. 3, pp. 267–283, 2012. [Online]. Available: http://ieeexplore.ieee.org/document/6224932/ 6

[74] H. Shin, D. Kim, Y. Kwon, and Y. Kim, "Illusion and dazzle: Adversarial optical channel exploits against lidars for automotive applications." [Online]. Available: http://eprint.iacr.org/2017/613.pdf 6

[75] J. Petit, B. Stottelaar, M. Feiri, and F. Kargl, "Remote attacks on automated vehicles sensors: Experiments on camera and lidar," *Black Hat Europe*, vol. 11, 2015. [Online]. Available: www.blackhat.com/docs/eu-15/materials/eu-15-Petit-Self-Driving-And-Connected-Cars-Fooling-Sensors-And-Tracking-Drivers-wp1.pdf 6

[76] C. Yan, X. Wenyuan, and J. Liu, "Can you trust autonomous vehicles: Contactless attacks against sensors of self-driving vehicle," *DEF CON*, 2016. [Online]. Available: http://dc.org/files/defcon24/Speaker%20Materials/DEFCON-24-Liu-Yan-Xu-Can-You-Trust-Autonomous-Vehicles-WP.pdf 6

[77] Y. Son, H. Shin, D. Kim, Y.-S. Park, J. Noh, K. Choi, J. Choi, Y. Kim *et al.*, "Rocking drones with intentional sound noise on gyroscopic sensors." in *USENIX Security Symposium*, 2015, pp. 881–896. [Online]. Available: https://www.usenix.org/system/files/conference/usenixsecurity15/sec15-paper-son.pdf 6

[78] Y. Shoukry, P. Martin, P. Tabuada, and M. Srivastava, "Non-invasive spoofing attacks for anti-lock braking systems," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2013, pp. 55–72. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-642-40349-1_4 6

Johann Thor Ingibergsson Mogensen received his B.Sc. and M.Sc. degrees as electronics engineer in Control and Automation from the Technical University of Denmark, in 2012 and 2014, respectively, and is currently employed at CLAAS E-Systems as an industrial Ph.D. student with affiliation to the University of Southern Denmark. His research interest covers robotics, domain-specific languages, and safety.

Dirk Kraft received his Diploma degree in Informatics from the University of Karlsruhe in 2006, and the Ph.D. degree in Robotics from the University of Southern Denmark, in 2009. From 2009 until 2013, he was Assistant Professor at the Maersk McKinney Moller Institute, the University of Southern Denmark where he is since 2013 an Associate Professor. His research interests lie within cognitive systems, computer vision, and robotics.

Ulrik Pagh Schultz received his B.Sc. and M.Sc. degrees in computer science from the University of Aarhus, in 1995 and 1997, respectively, and the Ph.D. degree in computing from University of Rennes, in 2000. From 2000 until 2005, he was a faculty member at University of Aarhus. Currently, he is an Associate Professor at the University of Southern Denmark. His research interest covers robotics, domain-specific languages, and software engineering. Since 2013 he is Chair of the IFIP Working Group 2.11 on Program Generation. He is a member of ACM and IEEE.