Increasing Safety by Combining Multiple Declarative Rules in Robotic Perception Systems

Johann Thor Mogensen Ingibergsson, Dirk Kraft, and Ulrik Pagh Schultz(✉)

Mærsk Mc-Kinney Møller Institute, University of Southern Denmark, Campusvej 55, 5230 Odense M, Denmark
{jomo,kraft,ups}@mmmi.sdu.dk

Abstract. Advanced cyber-physical systems such as mobile, networked robots are increasingly finding use in everyday society. A critical aspect of mobile robotics is the ability to react to a dynamically changing environment, which imposes significant requirements on the robot perception system. The perception system is key to maintaining safe navigation and operation for the robot and is often considered a safety-critical aspect of the system as a whole. To allow the system to operate in a public area the perception system thus has to be certified. The key issue that we address is how to have safety-compliant systems while keeping implementation transparency high and complexity low. In this paper we present an evaluation of different methods for modelling combinations of simple explicit computer vision rules designed to increase the trustworthiness of the perception system. We utilise the best-performing method, focusing on keeping the models of the perception pipeline transparent and understandable. We find that it is possible to improve the safety of the system with some performance cost, depending on the acceptable risk level.

Keywords: Safety · Computer vision · Robotics · Functional safety

1 Introduction

The significant growth of highly interconnected Cyber-Physical Systems (CPS) is currently imposing complex conglomerates of software and hardware on personal life and many sectors of industry. While the high degree of integration between software, mechanical, and electrical engineering is well-known and visible in sectors such as automotive or aviation, more "traditional" sectors such as agriculture and consumer electronics also benefit from the opportunities provided by the latest information and communications technology concepts and solutions. Driven by this trend, the domain of robotics is continuously expanding from large industrial machines in cages to free-moving consumer products. This expansion is reflected by the current market and projected increase in the future [11,22].

R. Chamberlain et al. (Eds.): CyPhy 2017, LNCS 11267, pp. 43–60, 2019.
https://doi.org/10.1007/978-3-030-17910-6_4

Computer vision is a key point for robotics—and thus CPS in general—to be able to act in a dynamic world [7]. The task for a computer vision system is to understand what exists, where a mobile robotic system is located, and if obstacles require an immediate action. These goals are functional requirements of the system and should be addressed with an explicit focus on safety when dealing with mobile robots. Indeed, a requirement for introducing autonomy in established domains is safety, which is done through compliance with functional safety standards that rely on code and documentation reviews. To support the use of vision in safety-critical systems, we propose to use simple and explicit computer vision rules as a means to determine specific problems in input images and the perception system as a whole [15]. Increasing the readability of the code and the perception rules can increase the overall trustworthiness of the system [13], facilitating the certification process [15]. Such simple and explicit computer vision rules can be implemented in a domain specific programming languages (DSL), supporting safe implementation of computer vision for safety-critical systems [14].

In this paper, we investigate how to combine such simple and explicit rules in an understandable way in order to judge the current operational safety of the robots perception system during autonomous operation. The safety system consists of code generated using the Vision Safety Language (ViSaL; [14]), where the developer specifies rules to safeguard the system from malfunctions. Our investigations build on the combination of explicitly written rules and address the issue of how this combination should be modelled, in particular, whether it can be done manually or automatically. The tested methods have a clear focus on being explicit and thereby easy to understand to facilitate clear communication to functional safety certifiers, such that the intent of the code and the system can be deducted. Concretely, we compare three manual programmatic methods with an automatic method based on decision trees. The decision tree method is deemed most appropriate in this comparison and is then assessed on a robot as an added safety layer for the You Only Look Once (YOLO) neural network (NN) [26] to investigate the performance and cost for the system. Compared to our earlier work where rules were evaluated independently [13–15], we here design, implement, and test combinations of rules and evaluate these combinations as a safety layer for a state-of-the-art NN.

The rest of this paper is structured as follows. In Sect. 2 we discuss safety in the context of robotics and related industries, followed by an overview of our initial work on safety and computer vision in connection to learning methods. In Sect. 3 we present different methods that can be used for modelling a safety layer for a computer vision system based on declarative rules, along with two datasets used for experimentally investigating the methods. The performance evaluation and comparison of the different methods will be conducted in Sect. 4, where they are tested on the datasets. The best-performing method is experimentally evaluated in Sect. 5, by introducing the safety method in a robotic system as a means to improve the safety of a NN. We end with an overall conclusion and an outline of future work in Sect. 6.

2 Fundamentals and Related Work

2.1 Safety in Robotics

While robotics is a thriving research area within academia, the penetration into industry has mostly been limited to replacing manual labour in factories. The intent of commercialising robots and selling them on a mass market puts certification as a central requirement as increasingly complex autonomous systems are introduced [27]. Certification allows products to comply with international standards, and thus lowers liability concerns [30]. A critical requirement for certification is that the software controlling the robot has been reviewed and paired with textual requirements, meaning that a clear interpretation of the intent is needed. Safety can be defined as, e.g., "*freedom from unacceptable risk*" [37]. Certification can consume more than 50% of the resources required to develop new safety-critical systems in related domains [1]; development of safety-critical software is expensive.

Safety is discussed in many settings and is often mentioned in papers, e.g., for computer vision, however without explicitly dealing with the challenges that arise [12]. As a result, safety is an obstacle for robots to operate autonomously in the public domain, while various ad-hoc safety measures have been designed, often focused on a specific risk such as collision, certification is not undertaken. It follows that there is a need for generalised methods for facilitating compliance and certification [20]. In the industrial domain developers rely on functional safety, which is defined as: "*part of the overall safety relating to the [equipment under control] EUC and the EUC control system that depends on the correct functioning of the [electrical and/or electronic and/or programmable electronic] E/E/PE safety-related systems and other risk reduction measures*" [37].

2.2 Perception Systems

The safety issue is partially at odds with how robotic autonomy is normally developed today. Researchers often rely on machine learning and Artificial Intelligence (AI) to achieve required performance and safety, however, the safety aspect is difficult for humans to interpret. This tension is described in *Defence Science Board 2016 Summer Study on Autonomy* which argues that it is hard for humans to understand and predict AI systems [6]. In computer vision, the focus is on performance [12], and the improved safety is often claimed despite introducing ever more complex algorithms. That "*in spite of their complexity, they fail frequently.* " moreover, "*in part **due to** their complexity, they fail in seemingly inexplicable ways.*" [2].

Robots rely on perception systems to navigate the world, as such the systems require both safety certification and a minimal performance. Safety certification could be done by adapting IEC 61496 [36] for outdoors use. Despite the scope of the standard being for machines in an industrial setting, it is noted that additional requirements can be applied to the system if it is intended for outdoor applications, which means that the developer has to argue for the safety in the system and thus can disregard the cages. Regarding performance, drafts exist of standards for outdoor robotics, e.g., ISO/DIS 18497 [35] and ISO/DIS 17757 [34]. The standards quantify detection performance, the tests have however been criticised [33]. The criticism is that the test object can be robustly detected in varying conditions using a NN, while not being able to detect humans.

The concept of safe states, i.e., fall-back behaviour, are important for perception systems since sensors, in general, have a risk of failing [3]. These failures require that the sensors are robust, without robustness the robot may "hallucinate" and respond inappropriately [23]. The failures are not limited to hardware, but also software and NNs "*need serious design effort, pre-planning and proved to be fragile in face of sensor errors in practice*" [38]. It is therefore important for functional safety to look at software safety verification of the input sensor data, and thereby to give assurance about the hardware by verifying inputs and outputs.

There are many requirements to a computer vision system [5]. Precision for vision algorithms is an issue in general, as a result it can be beneficial to create multiple classification regions. Multiple regions allow the system to deal with images that can be difficult to classify. Mekki-Mokhtar et al. proposed a concept called multiclass classification, as seen in other safety-critical systems [21]. The multiclass classification allows a system to have more than a boolean decision, e.g., "bad", "warning" and "good". This will enable a decision system to decide the trustworthiness of different sensors, and thereby decide if the robot needs to stop or just slow down. A benefit of using multiclass classification is tolerance towards the issue that sensors are in some way disturbed by noise, and thus the data may be unreliable. A common approach is to employ redundancy [10]. The multiclass classification can allow different sensors to use voting similar to sensor fusion [10].

A standard measure for evaluating algorithm performance within computer vision is Precision-Recall (PR). PR curves are used to identify the performance of classifiers. Other methods exists such as the well-known Receiver Operating Characteristics (ROC), however we choose PR curves over ROC because if a curve dominates in the PR-space, it will also dominate in the ROC space [4]. Furthermore, ROC does not take the baseline into account, and since categories for annotated data can be unbalanced, the PR-curves are a better statistical measure. An issue with the use of statistical measures is the notion of an acceptable risk, this is however not specified in functional safety standards. Therefore it is up to the developer to decide when the systems performance is good enough, with respect to acceptable risk. To evaluate the PR-curves we utilise Area Under the Curve (AUC). An approximation is however needed, such as trapezoidal approximation, to evaluate the AUC of the resulting PR curves and find an "optimum". This is because linear approximation is insufficient for PR [4]. The use of AUC in combination with PR instead of ROC is further supported by Saito et al. "*The PRC [Precision-ReCall] plot is more informative than ROC, CROC [Concentrated ROC], and CC [Cost Curve] plots when evaluating binary classifiers on imbalanced datasets*" [29]. With the hope of using these methods in connection with argumentation for functional safety, a perfect score would be desirable, however not a plausible result to aim for. This leaves the developer to rely on upcoming performance standards [34,35] and to assume an acceptable risk level for the system.

An issue with PR is however that one has two values that are being improved (precision and recall). To remedy this an optimum point is chosen based on the F_β *score*. The F_β *score* is a statistical measure that can be used to evaluate the classification performance. The β is a number reflecting a weight on either recall or precision. We have chosen 1 which results in the harmonic mean of precision and recall. The F_1 *score* is a weighted average where the best value is at 1 and the worst is 0.

2.3 Learning and Computer Vision

"*Software developers have a history of adding security to their products after the fact rather than integrating it into the development phase [...] Now, he warned, the machine learning community is poised to make the same mistake.*" [17]. We note that NNs 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 [18]. NNs should ideally be understandable and readable by humans, while still allowing for individual, meaningful rules [18]. Many industries have looked into the use of NNs [31], investigating the use of NNs since the 1980s [31].

Gupta et al. propose verification and validation of adaptive NNs [8], although with a focus on control systems. Specifically the absence of analytical certification methods restricts NNs to advisory roles in safety-related systems [19]. Moreover, we note that deep NN and similar recently popular methods share the same issue of being hard to assess in terms of safety. There do exist probabilistic measures for failure detection [40]. The issue with these methods is however that it is difficult to prove that the underlying distributions cover the entire normal behaviour. The difficulty is illustrated by the spurious behaviour learning methods can exhibit, where the neural network wrongly makes different classifications of images indistinguishable for humans [25]. The key problem is that classifiers and learning are complex tasks that are hard to prove reliable for humans, in particular through reviews. This shows that safety certification of NNs remains an open issue [9,19]. Because of functional safety and the requirement for code reviews, we do not believe that online trained or adaptive NNs are certifiable, despite recent advances [8].

To comply with functional safety the NN would have to be transparent. In the case of pre-trained NN the learning method and learned models would be available for the reviewer. The resulting NNs are however still "black boxes", moreover the computations can be time consuming [9]. This makes NN infeasible for safety-critical systems; however other learning methods exists that are more intuitive to understand, such as decision trees [24]. A decision tree is "*a hierarchical model composed of discriminant functions, or decision rules, that are* *applied recursively to partition the feature space of a dataset into pure, single-class subspaces*" [39]. A decision tree T consists of branches T_t where t is the number of branches. Decision trees can be explicitly depicted to give reviewers an intuitive understanding of the data flow. When using decision trees there is an inherent issue of over-fitting, which is addressed using pruning. Pruning refers to replacing branch nodes with leaves, thereby decreasing complexity and simplifying the tree. Trees can be trained with a stopping criterion, but it is generally accepted that it is better to overgrow a decision tree and then prune back [24]. "*Pruning a branch* T_t *from a tree* T *consists of deleting all descendants of t. Denote the pruned tree as* $T - T_t$*.*" [28]. There are multiple algorithms for pruning decision trees [28].

The NN for which we will introduce a safety layer in this paper is YOLO. From a software point of view, the safety check consists of executing code from ViSaL for assessing the input images for the YOLO algorithm, specifically we use YOLOv2 [26]. In this paper, we refer to YOLOv2 as YOLO. The algorithm finds anchor boxes of proposed predictions for the image, which are then thresholded based on a probability criteria. The predictions are then found using non-max suppression. Our reason for using YOLO is not the algorithm components, but its performance and speed, i.e., high precision and 45 frames per second [26].

Introducing a safety system could have a potential negative impact on the overall performance of the system, because some images categorised as good could be distrusted by the system. In a nutshell, a safety system that distrusts all images ensures that a NN will never make any miscategorisations, but is not useful. We use the term *uptime cost* to refer to the overhead of using a safety system, in terms of how many useful images are incorrectly discarded. Concretely, we measure the uptime cost by assessing how many images are removed from the highest category, i.e., the most usable images, to estimate the cost of introducing a safety layer to a perception system.

2.4 Programming Safe Perception Systems

Initial steps towards establishing a safe implementation of perception systems have been demonstrated using explicit declarative rules to address specific issues in a perception pipeline [14,15]. The rules are focused on different particularities of an image such as pixel distributions, pixel changes, and frequency. The simple rules use multiclass classification, because of the uncertainties in specifying precise thresholds for classifying images using the rules, thereby establishing a margin of reliability in the classifications. The many categories however also make the classification problem harder and thereby the performance of the algorithm deteriorates. As a result the concept of soft-boundaries was introduced to evaluate multiclass classification systems without penalizing the system performance excessively [15]. The computer vision rules have been implemented in the Vision Safety Language (ViSaL) DSL for automatic generation of their implementation [14] and an initial assessment has been performed with respect to readability with the goal of facilitating certification [13]. An excerpt of a ViSaL program is shown in Fig. 1, the rule FB detects images with abnormal distribution of pixels in a colour histogram, which for example detects underexposed or overexposed images unsuitable for further processing in a perception pipeline.

The declarative rules in ViSaL function independently, which means that ViSaL outputs individual scores for each rule, i.e., ok, warn or bad. The set of problems and symptoms that can result in problematic issues in a perception pipeline, are covered by a subset of the defined rules in ViSaL, therefore the entire rule set overlaps the problem space. This paper is focused on combining the rules, as this allows for a clear decision on the system integrity and can increase safety.

The focus of the combination of the rules is a means to give a clear statement on the integrity of the system, and as such the sum of rules does not in itself increase safety. Nevertheless, certain standards allow for combinations of safety-functions to increase safety. An example is ISO 26262 [16], where there exists a concept called decomposition, which allows for lower-rated safety function to safeguard high levels by combining safety-functions. In this paper the method of combining the rules is addressed, serving as an experimental continuation of the initial steps made in previously published papers [13–15].

```
...
Image(height=752, width=480) monoLeft = Bayer2Mono (source =
      camera.left.image);
Histogram h = histogram ( source = monoLeft, bins = 16 );
...
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;
 }
}
```

Fig. 1. Excerpt of ViSaL implementation of image analysis rules for verifying the data integrity of images [14].

3 Methods

3.1 Datasets

In our experiments, we use datasets consisting of images labelled using a usability category, where five is a usable image and one is unusable, i.e., unusable to make reliable safety decisions. Datasets consist of RGB and depth information and are processed by the rules ("raw data"), based on which thresholds are found in the data using precision-recall curves ("threshold data") [15]. We use two datasets: "agriculture", from an outdoor setting [15]; and "turtlebot" from an indoor setting. Both datasets consist of 406 images labelled using the usability category.

3.2 Decision Trees

We employ decision trees because we view them as a valid method in the context of functional safety, since the trees are an intuitively understandable visualization of a decision process. This means that the learned model can be verified and understood by tracing the propagation of images through the tree, and thereby iterate over an understandable model. Two decision trees are created using Matlab, where the trees are based on the raw data and on the threshold data respectively. We employ pruning on the automatically generated decision trees. For pruning we use cross-validation, meaning that the training data is split into train, test, and validation, to evaluate the model, i.e., the level of pruning for the decision tree.

3.3 Manual Programming

The manual programming approach utilises pre-defined thresholds for simple declarative rules, as to explicitly combine the rule evaluations, i.e., "bad", "warning" and "good" [15]. The use of a manual programming approach should allow the derived rules to be intuitively understood. Ultimately an extension of the ViSaL DSL would be used for this, but in this paper, we investigate the feasibility of such an approach, rather than the design of the language. For this reason, we use a mathematical notation to express the programmed rule combinations, leaving a concrete DSL syntax for rule combinations for future work. In this paper we consider three different manual programming approaches: "top-down","bottom-up", and "inferred learning".

The "top-down" approach refers to the idea that the combination of the rules are based on intuitive ideas by the developer, e.g., if all rules dealing with exposure or bin distributions are "good", then the image is assumed "good". The second approach,"bottom-up", relies on fitting a rule to the underlying data of the training set, e.g., the existing rule combinations. Last the "inferred learning" approach is based on creating rules using a combination of the two above approaches and by utilising insights manually inferred from an automatically generated decision tree.

The mathematical notation used to support the combination of rules can be seen in Fig. 2. It describes an overall evaluation of an image based on all rules R. The notation is based on a sequence S of compound rules P defined as propositions that combine rule evaluations $\alpha^r(x)$ using a weighting.

$$R \text{ set of all rules, } r \in R, I \text{ set of all images, } x \in I$$

$$\alpha^r(x) : \begin{cases} \mathbf{b}, & r(x) < t_r^{\text{error}} \\ \mathbf{g}, & r(x) > t_r^{\text{warn}} \\ \mathbf{w}, & \text{Otherwise} \end{cases}$$

$$\alpha_0^*(x) : \begin{cases} \mathbf{b}, & \exists r \in R : \alpha^r(x) = \mathbf{b} \\ \mathbf{g}, & \forall r \in R : \alpha^r(x) = \mathbf{g} \\ \mathbf{w}, & \text{Otherwise} \end{cases} \tag{1}$$

$$S \equiv \{(o_1, P_1), ..., (o_n, P_n)\}, \text{ where } o_i \in \{\mathbf{g}, \mathbf{w}, \mathbf{b}\} \tag{2}$$

$$\beta^S(x) : \begin{cases} o_m, & m = \min_{j \in \mathbb{N}} : P_j(x) \\ \alpha_0^*(x), & \forall m \in \mathbb{N} : \neg P_m(x) \end{cases}$$

Fig. 2. Mathematical representation of the combination of rules, where the thresholds t_r are assumed to always be an upper limit. The function $\alpha^r(x)$ corresponds to the evaluation of individual rules, where **b** corresponds to an image being categorised as "bad", **w** as "warning", and finally **g** as "good". The $\alpha_0^*(x)$ function is defined such that if one rule for a given image evaluates to "bad" then the combined result would be "bad", whereas if one image is"warn" and none are "bad", then the combined result is"warn". In all other cases, the result is "good", meaning that the rules have an equal weighting. The rule in Eq. 1 can be adapted for different scenarios. The use of the compound rules is done using a sequence S, consisting of an output and a compound rule, as described by Eq. 2. The sequence S of propositions is evaluated using $\beta^S(x)$, meaning that the first-occurring satisfied compound rule of an image will result in the images evaluating to the corresponding output value. If no compound rule is true then the result defaults to $\alpha_0^*(x)$.

Top-Down. The top-down approach relies on first combining the rules by using the dominant state of the rule results as in Eq. 1. Equation 1 implies an equal weighting of the rules. The combination of rules is created as compound rules, meaning that they are designed as propositions P combined in a sequence S, and evaluated using $\beta^S(x)$.

Bottom-Up. The bottom-up approach uses the same initial combination approach (Eq. 1). The compound rules for this task are found by investigating the data, e.g., using a loop for testing all possible rule combinations and their performance. The best-performing combinations are then introduced, and the process can be iterated. An alternative approach used is to investigate the existing rule combinations in the data. The new rules are therefore found based on the chosen training data.

Inferred Learning. Because of the risk of over-fitting the inferred learning approach utilizes pruned decision trees. Using decision trees as heuristics for creating manual rules could possibly benefit the creation. The key benefit of using the heuristics is however that the decision tree and in particular the predictor importance allows for improving the weights in $\alpha_0^*(x)$ (Eq. 1) because there is an understanding of the rule's impact based on the available data.

4 Combining Declarative Rules

4.1 Combination of Rules

We benchmark the three manual approaches (top-down, bottom-up, and inferred learning) and the two decision trees (non-pruned and pruned). A key issue is that manual rules are only created once per split, meaning that the statistical properties such as the mean and the standard deviation are impacted since some training data will be present in the test data. We however believe that it is still interesting to see the statistical results as a means to evaluate the prediction methods with respect to the decision trees.

For functional safety the optimal process for conveying information would be to create manual rules based on expert knowledge, as to argue for the logic behind the choices, which allows for explicitly pairing the safety goals to the implementation, thereby making a clear reference for the reviewer. As an example we show an extract of the rules defined as a result of the top-down (TD) analysis, where the three ViSaL rules $CAbot$, $CAtop$, and BF are used. These rules deal with connected components in the image ($CAbot$ and $CAtop$) whereas BF finds the largest bin and its corresponding fill level.

$$\begin{aligned}
P_{g_1}^{TD}(x) &\equiv \exists r' \in R \setminus \{CAbot, CAtop, BF\} : \\
&\quad \alpha^{CAbot}(x) = \alpha^{CAtop}(x) = \alpha^{BF}(x) = \alpha^{r'}(x) = \mathbf{g} && (3)\\
P_{b_1}^{TD}(x) &\equiv \alpha^{CAbot}(x) = \alpha^{CAtop}(x) = \mathbf{w} && (4)\\
P_{b_2}^{TD}(x) &\equiv \alpha^{FR}(x) = \mathbf{w} \wedge \left(\alpha^{CAbot}(x) = \mathbf{w} \vee \alpha^{CAtop}(x) = \mathbf{w}\right) && (5)\\
P_{b_3}^{TD}(x) &\equiv \exists r' \neq r'' \neq r''' \neq r'''' \in R : \\
&\quad \alpha^{r'}(x) = \alpha^{r''}(x) = \alpha^{r'''}(x) = \alpha^{r''''}(x) = \mathbf{w} && (6)
\end{aligned}$$

The propositions P shown in Eqs. 3 to 6 are examples of the rules found using the top-down analysis. The top-down manual rules were found by first combining the rules individual results with $\alpha_0^*(x)$. This results in the combination represented as a sequence S shown in Eq. 7.

$$S_{TD} \equiv \{(\mathbf{b}, P_{b_3}^{TD}), (\mathbf{b}, P_{b_2}^{TD}), (\mathbf{b}, P_{b_1}^{TD}), (\mathbf{g}, P_{g_1}^{TD})\} \qquad (7)$$

The performance of the manual approaches top-down and bottom-up can be seen in Table 1, where it is evident that they are not feasible. The inferred learning approach which uses the decision tree as heuristics for creating the manual

Table 1. The mean, μ, and the standard deviation, σ, of the individual results of the pruned decision tree training for the two datasets, on the raw and threshold results on the 200 random splits (see also Sect. 3.1).

		Turtlebot test pruned	Agriculture test pruned
Decision trees	threshold	μ 0.40 σ 0.32	μ 0.33 σ 0.07
	raw	μ 0.95 σ 0.01	μ 0.80 σ 0.02

		Top-down		Bottom-up		Inferred learning	
		μ	σ	μ	σ	μ	σ
Manual programmatic	Agriculture	0.80	0.03	0.80	0.03	0.80	0.03
	Turtlebot	0.83	0.02	0.78	0.03	0.91	0.02

rules has better performance and the combination of rules was also done faster by the human programmer. Nevertheless, the results of the decision trees still outperformed the manually programmed rules. It should not be concluded that manually programmed rules can be disregarded, but rather that the overall performance can be improved using decision trees. Intuitive rules can still be created to express specific safety goals, assuming the performance is good enough. The methods should, therefore, be viewed as complementary with respect to complying with safety standards. Nevertheless the decision trees significantly outperformed the manually programmed rules, as can be seen in Table 1, where it is evident that the "raw" results outperform the"threshold" results.

5 Experimental Evaluation

5.1 Robot Platform

For our experimental evaluation, we utilise a robot from Conpleks Innovation ApS, shown in Fig. 3. The robot runs ROS Kinetic on Ubuntu 16.04, and the system navigates autonomously using an RTK-GPS and perception sensors. The software for the robot runs on two embedded platforms. First, the controller for the motors is a Conpleks robotech controller 501 based on an Aaeon GENE-QM77, which is an embedded single-board computer consisting of a 3rd generation Intel i5 processor. Second, the sensor fusion platform is based on a GeForce GTX 1080 interfaced to an Intel i7-6700K, which enables real-time processing of the sensor data. This robot is equipped with a sensor kit consisting of a thermal camera, a stereo camera, and a 360 degrees lidar. The current sensor fusion platform is built using consumer-grade electronics and is as such not usable for an industrial setting. For data collection, the robot was controlled manually and only the stereo camera was used.

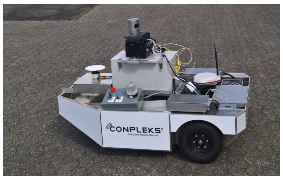


Fig. 3. Conpleks' robot created for collecting golf balls autonomously [32].

5.2 Test Setup

The recordings are used to evaluate the combination of the YOLO algorithm and the safety layer. The evaluation is thus not performed on-line during operation of the robot, but at a later time using recorded images. We implemented YOLO on industrial-grade embedded boards, namely NVidia Jetson TX2 boards. The Jetson TX2 board was used as the test setup and is running ROS Kinetic on Ubuntu 16.04, making it suitable future for deployment on our robotic platform.

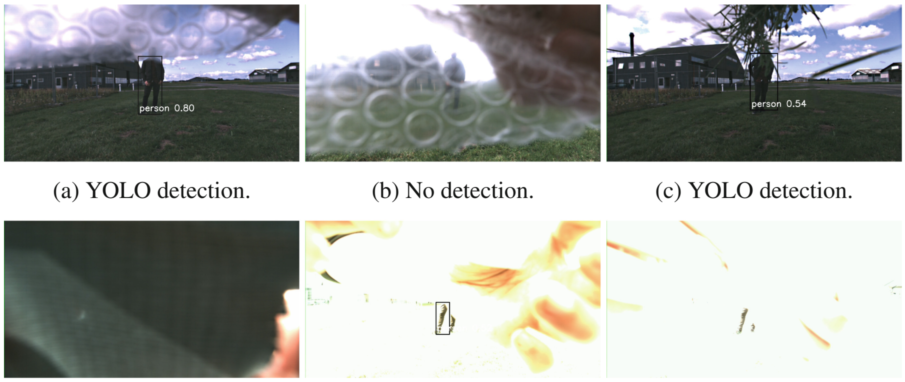
5.3 Data Acquisition

The recordings of the dataset were done in Odense Airport 21st-22nd of August 2017. The recording session consisted of an operator remote controlling the robot while between zero and two persons moved around in front of the robot. In addition, a second part of the recordings consisted of stressing the camera by exposing it to potential hazards for the image: over- and under-exposure was simulated by emitting light into the lens and covering it up; frozen image was simulated by covering the lens; dirty lens was simulated using grass, dirt, and water. Frozen image normally means that the same image is emitted several times, i.e., it would not be limited to black images. This means that the experiment is a special case. Nevertheless, the general example would be caught by rules using optical flow and/or lack of pixel changes, both of which are examples of rules which we use [15]. In addition, the focus was changed by adding different levels of water and plastic in front of the lens to distort the scene.

5.4 Dataset

The recordings consisted of 4471 images which are manually labelled based on the usability category. One researcher (the first author) did the labelling manually. Randomised images for the five usability categories were then extracted with an equal weighting for all categories for the data analysis. The equal weighting allows us to create three datasets: training; test; and validation, where each set consists of 545 images, which were split evenly into the five categories, i.e., 109 images per category.

The dataset is further annotated based on the YOLO algorithms performance, this is done after the usability categorisation, as to not have an impact



(a) YOLO detection. (b) No detection. (c) YOLO detection.

(d) No detection. (e) YOLO detection. (f) No detection.

Fig. 4. Image examples from the dataset evaluated by the YOLO algorithm, where the bounding boxes indicate if the YOLO algorithm made a detection. Images a–d had their brightness increased similarly to help the reader see the structures in the images, whereas images e and f retain their original brightness.

on the results. The YOLO labels are 1 if there is at least one human present, and 0 if no persons are present in the image. The detection precision of YOLO is also labelled by 1 and 0.

5.5 Initial Data Exploration

To investigate if the labelling was acceptable we examine it using the YOLO detections as a baseline. From the entire dataset of 4471 images, there are 1942 good images with people in the scene. Out of these, the YOLO detected at least one human in 1922. Resulting in a failure percentage of 1.04%. Second, we look at the bad images consisting of 607 images with people in the scene. YOLO was able to detect a human in 405 images. Resulting in a failure percentage of 33.28%. We use these results as an indication that the usability labelling of the images seems to be correct. Despite YOLO detecting correctly in 66.7% of the bad images, they can still be interpreted as bad, referring to Fig. 4. This means that a correct detection is not the same as stating the image is usable.

5.6 Usability

The results are extracted based on 192 random splits used for both the pruned and non-pruned decision trees. The results for the precision, recall, and F1-scores for the decision trees can be seen in Table 2.

We test the impact of the pruning using the F1-score via a paired t-test. We do this for both the test and validation splits where we test the null hypothesis that the pairwise difference between data vectors of the pruned decision tree (x) and non-pruned decision tree (y) has a mean equal to zero at the 1%

Table 2. The precision, recall and F1 scores for the *non-pruned* and *pruned* decision trees evaluated on the 192 random splits in the dataset.

Non-pruned		Precision	Recall	F1
test	mean	0.8911	0.8384	0.8636
	std	0.0165	0.027	0.0134
Valid	mean	0.8852	0.8447	0.8642
	std	0.0264	0.0218	0.0186

Pruned		Precision	Recall	F1
test	mean	0.9429	0.8741	0.907
	std	0.0211	0.0224	0.0176
Valid	mean	0.941	0.8728	0.9054
	std	0.0263	0.0197	0.0177

significance level. We evaluate this against the alternative hypothesis that the pruned decision tree (x) has a greater mean than the non-pruned decision tree (y) with a 1% significance level. The null hypothesis is rejected, meaning that the pruned decision trees have a higher performance on unknown data, compared to the non-pruned. Comparing the results with the manual results on the older datasets in Table 1, it is evident that the new decision trees perform better. We use these comparisons to conclude that the decision trees are applicable.

5.7 Assessment

The decision trees based on the ViSaL rules are now used to analyse the images before the YOLO algorithm is applied. The goal of the system is to signal to a decision system whether or not the data is trustworthy. This means that if the image is not usable or if there is a human in the image, the robot should be stopped. We, therefore, conduct three analyses for comparison. The three different analyses are: using only the YOLO algorithm without any impact from the ViSaL rules; using YOLO and ViSaL based decision trees, not trusting bad images; and finally YOLO and ViSaL based decision trees only trusting good images. These assessments are done for both the pruned decision tree and can be seen in Table 3.

In Table 3 the rows represent the results based on 192 iterations. The first row "Total Images" corresponds to the number of images that is checked in every iteration. The second row "Trusted Images" refers to the images that YOLO is exposed to, the range is a result of the random sampling for the 192 iterations. The "Humans Missed" row refers to the humans that are visible in the image and that were not detected by YOLO. "Mean Percentage Detection" refers to the detection performance of the YOLO algorithm on "Trusted Images", where if at least one human was detected it was a correct detection. The "Uptime Cost" row is based on the usable images in the fifth category and how many of those have been removed by the ViSaL decision trees. The cost is only based on the fifth category because we are using the soft boundary method [15], which means that the fourth category can be moved into the warning region. Finally, the last two rows give the F1 scores of the YOLO detections and the safety layer. For the YOLO detections it is calculated based on: a true positive is a correct detection of all humans in the image; a false positive is a wrong detection in the image; false negative is if YOLO misses a detection in the image; and true negative is if

Table 3. The table illustrates the performance impact of using ViSaL-based pruned decision trees in connection with YOLO, see text for details.

	YOLO no safety		YOLO, distrust "warn" and "bad"		YOLO, distrust only "bad"	
	mean	std	mean	std	mean	std
Total images	545	0	545	0	545	0
Trusted images	545	0	112.56	9.34	328.63	9.10
Humans missed	62.25	7	2.63	1.62	11.38	3.11
Percentage detection	88.58%	0.0128	97.67%	0.0139	96.54%	0.0096
Uptime cost	0%	0	28.78%	0.0521	4.99%	0.0225
F1 for detections	0.82	0.0136	0.87	0.0337	0.88	0.0154
F1 of safety layer	N/A	N/A	0.88	0.0380	0.92	0.0159

no humans are present and there are no detections. Furthermore, it is important to note that although the number of images removed seems excessive, the two worst categories consist of 218 images. This means that when the system only distrusts bad images, the trusted images column will decrease with around 218 images, as can be seen on the right side of the table. For the middle column, the number of removed images increases because there potentially are 327 warning images due to the soft boundaries, where the warning region is incorporated.

From Table 3 it can be seen that the removed images correspond to the intuition from earlier. In addition, it is evident that the introduction of the safety layer drastically reduces the number of missed persons. This means that the choice of the aggressiveness of the safety layer, i.e., the reduction of accepted images, has to be evaluated as to what is an acceptable and an unacceptable risk.

6 Conclusion and Future Work

In this paper, we investigated how to model the combination of declarative rules as a means to improve a vision pipeline with respect to performance and safety. The modelling choices focused on a standard approach to learning within computer vision. The data is recorded using a robot platform and analysed on an industrial-grade embedded board to have a feeling for real-time performance. We found that introducing the safety layer into a vision system can improve the performance, with some uptime cost. The definition of acceptable risk is, therefore, a critical issue for fielding autonomous systems that rely on approaches where human operators are not in the decision loop.

To use decision trees in functional safety it is critical to understand the intent and to generate code for embedded platforms, we believe that such an approach can improve the communication with certification authorities [39]. This means that understandability and readability are equally critical for the software. While readability for ViSaL has been investigated [13], further studies are needed. Finally, a more detailed analysis of specific faults is needed to understand which specific hazards are currently detected robustly and which need more rules.

References

1. Baheti, R., Gill, H.: Cyber-physical systems. Impact Control Technol. **12**, 161–166 (2011)
2. Bansal, A., Farhadi, A., Parikh, D.: Towards transparent systems: semantic characterization of failure modes. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8694, pp. 366–381. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10599-4_24
3. Daigle, M.J., Koutsoukos, X.D., Biswas, G.: Distributed diagnosis in formations of mobile robots. IEEE Trans. Robo. **23**(2), 353–369 (2007)
4. Davis, J., Goadrich, M.: The relationship between precision-recall and roc curves. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 233–240 (2006)
5. De Cabrol, A., Garcia, T., Bonnin, P., Chetto, M.: A concept of dynamically reconfigurable real-time vision system for autonomous mobile robotics. Int. J. Autom. Comput. **5**(2), 174–184 (2008)
6. Fields, C., David, R., Nielsen, P.: Defense science board 2016 summer study on autonomy. Defense Science Board (2016)
7. Frese, U., Hirschmüller, H.: Special issue on robot vision: what is robot vision? J. Real-Time Image Process. **10**(4), 597–598 (2015)
8. Gupta, P., Loparo, K., Mackall, D., Schumann, J., Soares, F.: 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)
9. Hauge, A., Tonnesen, A.: Use of artificial neural networks in safety critical systems. Faculty of Computer Sciences (2004)
10. Heckemann, K., Gesell, M., Pfister, T., Berns, K., Schneider, K., Trapp, M.: Safe automotive software. In: König, A., Dengel, A., Hinkelmann, K., Kise, K., Howlett, R.J., Jain, L.C. (eds.) KES 2011. LNCS (LNAI), vol. 6884, pp. 167–176. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23866-6_18
11. IFR: World Robotics 2014 Industrial Robots (2014)
12. Ingibergsson, J.T.M., Schultz, U.P., Kuhrmann, M.: On the use of safety certification practices in autonomous field robot software development: a systematic mapping study. In: Abrahamsson, P., Corral, L., Oivo, M., Russo, B. (eds.) PROFES 2015. LNCS, vol. 9459, pp. 335–352. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26844-6_25
13. Ingibergsson, J.T.M., Hanenberg, S., Sunshine, J., Schultz, U.P.: Readability study of a domain specific language: process and outcome. In: Accepted for the 33rd ACM/SIGAPP Symposium on Applied Computing (SAC-18) (2018)
14. Ingibergsson, J.T.M., Kraft, D., Schultz, U.P.: Declarative rule-based safety for robotic perception systems. J. Software Eng. Rob. (JOSER) **8**(1), 17–31 (2017)
15. Ingibergsson, J.T.M., Kraft, D., Schultz, U.P.: Explicit image quality detection rules for functional safety in computer vision. In: 12th International Conference on Computer Vision Theory and Applications (VISAPP), p. 12, Marts 2017

16. ISO TC22/SC3/WG16. ISO/IEC 26262:2011: Road vehicles - Functional safety. Technical report, International Organization for Standardization (2011)
17. Klarreich, E.: Learning securely. Commun. ACM **59**(11), 12–14 (2016)
18. Kurd, Z., Kelly, T.: Establishing safety criteria for artificial neural networks. In: Palade, V., Howlett, R.J., Jain, L. (eds.) KES 2003. LNCS (LNAI), vol. 2773, pp. 163–169. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45224-9_24
19. Kurd, Z., Kelly, T., Austin, J.: Safety criteria and safety lifecycle for artificial neural networks. In: Proceedings of Eunite, vol. 2003 (2003)
20. Machin, M., Dufossé, F., Blanquart, J.-P., Guiochet, J., Powell, D., Waeselynck, H.: Specifying safety monitors for autonomous systems using model-checking. In: Bondavalli, A., Di Giandomenico, F. (eds.) SAFECOMP 2014. LNCS, vol. 8666, pp. 262–277. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10506-2_18
21. Mekki-Mokhtar, A., Blanquart, J.-P., Guiochet, J., Powell, D., Roy, M.: Safety trigger conditions for critical autonomous systems. In: 18th Pacific Rim International Symposium on Dependable Computing, pp. 61–69. IEEE (2012)
22. METI: Trends in the Market for the Robot Industry in 2012, July 2013
23. Murphy, R.R., Hershberger, D.: Handling sensing failures in autonomous mobile robots. Int. J. Robot. Res. **18**(4), 382–400 (1999)
24. Myles, A.J., Feudale, R.N., Liu, Y., Woody, N.A., Brown, S.D.: An introduction to decision tree modeling. J. Chemom. **18**(6), 275–285 (2004)
25. Nguyen, A., Yosinski, J., Clune, J.: Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In: Conference on Computer Vision and Pattern Recognition (CVPR), pp. 427–436. IEEE (2015)
26. Redmon, J., Farhadi, A.: YOLO9000: Better, faster, stronger. arXiv preprint arXiv:1612.08242 (2016)
27. Reichardt, M., Föhst, T., Berns, K.: On software quality-motivated design of a real-time framework for complex robot control systems. In: International Workshop on Software Quality and Maintainability (2013)
28. Safavian, S.R., Landgrebe, D.: A survey of decision tree classifier methodology. IEEE Trans. Syst. Man Cybern. **21**(3), 660–674 (1991)
29. Saito, T., Rehmsmeier, M.: The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. In: PLoS ONE, pp. 1–21 (2015)
30. Santosuosso, A., Boscarato, C., Caroleo, F., Labruto, R., Leroux, C.: Robots, market and civil liability: a european perspective. In: RO-MAN, pp. 1051–1058. IEEE (2012)
31. Schumann, J., Gupta, P., Liu, Y.: Application of neural networks in high assurance systems: a survey. In: Schumann, J., Liu, Y. (eds.) Applications of Neural Networks in High Assurance Systems, pp. 1–19. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-10690-3_1
32. SDU: Marken er mejet af en robot (2017)
33. Steen, K.A., Christiansen, P., Karstoft, H., Jørgensen, R.N.: Using deep learning to challenge safety standard for highly autonomous machines in agriculture. J. Imaging **2**(1), 6 (2016)
34. TC 127: Earth-moving machinery - autonomous machine system safety. In: International Standard ISO 17757–2015, International Organization for Standardization (2015)
35. TC 23: Agricultural machinery and tractors - Safety of highly automated machinery. International Standard ISO/DIS 18497, International Organization for Standardization (2014)

36. TC 44: Safety of machinery - electro-sensitive protective equipment. International Standard IEC 61496–2012, International Electronical Commission (2012)
37. TC 65: Safety of machinery - electro-sensitive protective equipment. International Standard IEC 61508–2011, International Electronical Commission (2011)
38. Veres, S.M., Lincoln, N.K., Molnar, L.: Control engineering of autonomous cognitive vehicles-a practical tutorial. Technical report, Faculty of Engineering and the Environment, University of Southampton, Technical report (2011)
39. Yang, Y., Keller, P., Livnat, Y., Liggesmeyer, P.: Improving safety-critical systems by visual analysis. In: OASIcs-OpenAccess Series in Informatics, vol. 27. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2012)
40. Zhang, P., Wang, J., Farhadi, A., Hebert, M., Parikh, D.: Predicting failures of vision systems. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3566–3573 (2014)