Regular Article

Perception robustness testing at different levels of generality

Zachary Pezzementi[1], Trenton Tabor[1], Jonathan K. Chang[1], Carter Tiernan[1], Bill Drozd[1], Eric Sample[1], Chris Hazard[1], Michael Wagner[2] and Philip Koopman[3]
[1]National Robotics Engineering Center (NREC), Carnegie Mellon University, Pittsburgh, Pennsylvania, USA
[2]Edge Case Research LLC, Pittsburgh, Pennsylvania, USA
[3]Electrical and Computer Engineering Department, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA

Abstract: Ensuring the success and safety of deployed robotic systems often requires predicting their behavior in a wide range of conditions, precluding practical testing of the physical system itself. We introduce an approach to predict the robustness of a perception system in a broad range of conditions through simulation of challenging real-world conditions achieved by modifying real images according to physical phenomena, producing physically-realistic, mutated images. Typical perception-system performance metrics are difficult to relate to system-level application- or safety-requirements. We propose performance-evaluating metrics that are more closely tied to these targets, a relationship dependent on the amount of application-specific context available and assumptions imposed. We also introduce a novel robustness metric, Robustness ROC, that can be applied to any of these performance metrics. We demonstrate this method on NREC-AgPD, a large dataset focused on person detection for off-road driving in orchard environments, assessing several recent person detectors, and showing how considerations of robustness can often change the best-choice algorithm.

Keywords: computer vision, learning, perception

1. Introduction

The rapid growth of autonomous systems being developed for safety-critical applications requires efficient and effective methods to test and ensure these systems will be robust in the wide range of situations they will encounter. Ensuring safety through exhaustive real-world testing is infeasible for most practical systems, requiring on the order of billions of test miles or hours of autonomous operation for a large-scale autonomous vehicle fleet (Kalra & Paddock, 2016; Koopman & Wagner, 2016). A promising alternative is to make use of the ever-growing availability of computational resources to expand upon what is feasible with real-world testing. It is often infeasible to field test

This work was performed while Michael Wagner was at NREC.

Received: 29 September 2020; revised: 21 April 2021; accepted: 25 April 2021; published: 01 December 2021.

Correspondence: Zachary Pezzementi, National Robotics Engineering Center (NREC), Carnegie Mellon University, 10 40th Street, Pittsburgh, PA 15201, USA, Email: pez@nrec.ri.cmu.edu

every algorithm under consideration for a problem, so it is important to have techniques to guide the process of algorithm selection and parameterization before deployment.

A promising avenue of previous work (discussed further in Section 2.1) modifies real images by applying mutations that model degrading effects in order to evaluate systems' robustness to these effects. This approach allows developers to augment previously logged runs of the robot system with effects that would normally require much more data collection to capture, greatly lowering the effort to achieve test coverage breadth. In this work, we introduce techniques to evaluate the performance and safety of a perception component early in the design process, while taking into account its robustness to a variety of challenging conditions. While some prior work first demonstrated the use of simulated but physically-realistic, adverse conditions that augment real data to measure robustness (Pezzementi, Tabor, Yim, et al., 2018), this is the first work to measure robustness in terms of the performance of the robotic system of which the perception module is a part, and it introduces improved metrics to recognize vulnerabilities missed by prior work. During the development of integrated robotic systems, many component-level choices must be made before the system is fully embodied. We propose evaluation methods to guide algorithm selection at this stage while remaining largely agnostic to algorithm parameter choices, but allowing for incorporation of available application-specific context.

Drawing from and building on existing work in simulation and robustness testing (discussed further in Section 2), we assemble a set of realistic image mutations that represent a variety of challenging conditions that are often absent from training and evaluation data, but are conditions that a robotic perception system may encounter when deployed. We then apply them to a large-scale dataset, NREC-AgPD (Pezzementi, Tabor, Hu, et al., 2018), a dataset of stereo videos (nearly 100K labeled frames) from robotic orchard vehicles primarily driving up to people in many scenarios, with variation in appearance, pose, activity, level of occlusion, and setting. This allows us to evaluate the effects of the mutations on performance for a safety-critical task, person detection. The mutations applied include procedural mutations such as blurring and changes that remove image data, as well as what we call "contextual mutations" that make use of environment geometry information to perform depth-based simulation of haze and defocus effects.

We evaluate robustness across one dimension in the space of algorithm parameterizations through the use of trade-off curves and the changes imparted on them by these image mutations. Our approach is inspired by the receiver operating characteristic (ROC) curve, which is often used in problems such as classification and detection to characterize the trade-off between true and false detections by evaluating all possible combinations achievable by varying an underlying sensitivity threshold. We demonstrate how to generalize this concept to analyze how varying one parameter affects overall system performance with respect to, for example, safety invariants. We do this by introducing additional levels of system and application context to generate different trade-off curves. Then we show a new way to compare curves resulting from images with and without mutations, to compute robustness to those mutations. The technique, which we call Robustness ROC (RobROC, see Table A1 for list of key symbols and acronyms), also improves on shortcomings of traditional ROC analysis in the context of evaluating robustness: We will show that it captures robustness problems that previous metrics improperly ignore.

Optimizing a system for safety involves an inherent trade-off with its availability/efficiency/productivity. At one extreme, one can guarantee a vehicle will not carry out unsafe motions by preventing all motion, but this prevents it from being able to do anything productive. More practically, decisions of whether to deploy a robot to automate a process are usually driven by projections of its effect on efficiency, so it is essential to keep track of efficiency simultaneously with optimization to achieve the desired level of safety. Multiple-Criteria Decision Making (MCDM) provides techniques for analyzing such problems with multiple, potentially-conflicting goals and is discussed further in Section 2.4.

We apply our technique on an example application, namely person-detection to safeguard an autonomous orchard robot, exploring the detection-sensitivity parameter space. By evaluating a variety of different algorithms' performance, we show that performance rankings can change under these effects; i.e., a system designer's choice of the best algorithm for the task would change if robustness to these effects is important, compared to the standard analysis that does not consider robustness. We also explore the effects of introducing different levels of application and system context (and corresponding assumptions and requirements on the analysis) for this problem and how well trends generalize across context levels. In our example application, this context captures the idea that it's much more important to detect people immediately in the vehicle path than those far from it; we find this additional context can also change decisions, showing value in performing analysis with as much context as possible.

Our key contributions are therefore:

- Robustness ROC, a novel method to measure robustness to unusual conditions, applicable to a general class of trade-off curves
- Proposals of possible trade-off curves for a robotic safeguarding application, incorporating varying amounts of application context to facilitate tying performance back to system requirements
- Application of our Robustness ROC technique to a varied set of person detectors on a large dataset, along with analysis of robustness trends with respect to diverse, realistic image mutations

2. Related work

2.1. Robustness

Some previous work has investigated performance degradation from simple image processing effects or "image mutations", usually on the task of classification/recognition (and not the more complex task of detection). Most notable among this work is the creation of the ImageNet-C dataset (Hendrycks & Dietterich, 2019), which models 15 types of image corruption at different levels of severity. About half of these model simple processes including noise (Gaussian, shot, and impulse), basic image processing (brightness and contrast), and compression artifacts (pixelate, JPEG). The rest are based on a variety of physical phenomena, such as fog, snow, defocus and other types of blur; since they do not have 3D information available though, they can not be modeled with high fidelity. Follow-on work finds that humans are much more robust than deep networks to most mutations, though augmenting on one type of mutation helps with robustness against that particular mutation (often giving better robustness than humans) (Geirhos et al., 2018). Unfortunately it offers only limited robustness gains against other mutations and attacks (Geirhos et al., 2018; Gu et al., 2019; Kang et al., 2020), and it is infeasible to train on all combinations of mutations. AutoAugment provides a method to learn an augmentation strategy that best fits a target dataset (Cubuk et al., 2019), and AugMix adds a consistency objective across mutations (Hendrycks et al., 2020).

DeepXplore (Pei et al., 2017) evaluates robustness on an end-to-end learning task, using the DAVE-2 self-driving system. Since ground truth correct answers are not available, they use consensus between systems as a proxy for an oracle of the desired output. They then introduce the idea of neuron coverage of test inputs to whitebox DNNs as a metric for whether a system's robustness is being effectively evaluated and propose it as improvement over code coverage. However, other work has found this method to be an ineffective indicator of robustness (Dong et al., 2020).

VISTA (Amini et al., 2020) uses real image data with stereo depth maps and interpolates them to realistically estimate images from novel viewpoints to train reinforcement learning approaches.

Evaluating robustness of object detectors is a recent area of research. Michaelis et al. (2020) applies mutations that include a mix of simple simulation of physically realizable conditions, such as snow, and other less realistic conditions, such as elastic transformations, and they evaluate robustness of object detection on Pascal VOC, COCO, and CityScapes. In our own previous work, Pezzementi, Tabor, Yim, et al. (2018), we considered only physically realistic mutations, which we again apply here. Others have begun to make use of depth information to do more physically realistic modeling of several weather phenomena. Using fog models very similar to ours, Foggy CityScapes (Sakaridis et al., 2018) generates fog-perturbed images of the CityScapes dataset, and von Bernuth et al. (2019) shows robustness results on the KITTI object detection benchmark, along with physically-realistic snow modeling. Volk et al. (2019) similarly evaluates performance against a physically-realistic rain model on the KITTI object detection benchmark, with an iterative retraining scheme to achieve robustness. Additionally, Zhang and Wang (2019) examined how to improve robustness of detection with respect to adversarial mutation. However, all of this work follows the trend of evaluating robustness by simply comparing average precision before and after perturbing the image set, which we will show does not capture many changes in behavior that may result for a deployed system. Zhong et al. (2020) uses a similar set of mutations to Michaelis et al. (2020) and proposes a robustness metric that measures the minimum mutation (in the sense of L_p norm) required to lead to an error, averaged across all the evaluation images; we argue that this metric is less extensible to understanding the magnitude of effects at the system level though.

2.2. Simulation

Another popular tool for selecting algorithms is using simulation, with many available datasets, such as the public Virtual KITTI 2.0 (Gaidon et al., 2016) and SYNTHIA (Johnson-Roberson et al., 2017), to provide testing environments. While it has been shown that performance of models in simulation can be quite different than in the real world, the performance trends and failure modes of models still hold (Veeravasarapu et al., 2015). Higher fidelity physical modeling can be a solution here and is described by the various major, commercial, autonomous car developers (Argo AI, 2019; Baidu AI, 2017; Bigelow, 2019; Madrigal, 2017; Nvidia, 2018; Reynolds, 2019), but can require significant resources and effort to capture the variation of the real world adequately, which is important for algorithms that are looking for complex patterns.

That level of realism can also be achieved without as much developmental scope or simulation expertise by starting from real images and augmenting them with simulated physical effects. This approach can also show unique characteristics of the target sensor. This type of simulation shows significant performance improvements over more traditional augmentation techniques (Volk et al., 2019) or normalization methods, such as dehazing (Sakaridis et al., 2018). More accurate physical modeling has also enhanced other vision tasks such as 3D dense registration (Meilland et al., 2013)

2.3. Adversarial

Another concern explored in recent papers is whether adversarial methods can be applied in the real world to produce objects that would be misinterpreted by the robot. Researchers have found deep networks to be quite susceptible to adversarial attacks due to batch normalization (Galloway et al., 2019) and inherent properties in datasets (Ilyas et al., 2019). Kurakin et al. (2018) showed that printing out adversarial mutations of images still results in misclassification. Eykholt, Evtimov, Fernandes, Li, Rahmati, et al. (2018) and Eykholt, Evtimov, Fernandes, Li, Song, et al. (2018) took this further and showed its robust outdoor applicability to stop sign classification. Athalye et al. (2018) then showed that 3D printed objects could have the same effects from a wide range of views. Wu et al. (2020) also considers adversarial attacks that can be physically realized and proposes a training approach to increase robustness, though only to some classes of such attacks. Furthermore, training on adversarial examples in an attempt to be robust to attacks tends to reduce accuracy on the nominal performance and increase training time (Raghunathan et al., 2019).

However, these attacks are most successful on classification networks or when white-box access to the network is available, but not as successful or transferable to networks for detection without internal model access. In addition, naturally-occurring phenomena pose a much larger obstacle to fielding robotic systems. Pezzementi, Tabor, Yim, et al. (2018) showed examples of situations where nearly imperceptible natural image modifications can still result in dramatic perception changes.

2.4. Multiple Criteria Decision Making

One field we take inspiration from is Multiple-Criteria Decision Making (MCDM). In MCDM there are techniques for analyzing problems with multiple design variables and, notably, multiple criteria or goals. These analyses readily acknowledge that goals may be in conflict. For an autonomous vehicle, the safest course is to stay in place; the most productive is to rush headlong along one's path. One basic concept from MCDM that has an easy mapping to robotics is that decision making has two spaces of interest: the decision space and the objective space. These can be easily mapped to the robotics concepts of the configuration and task space. The decision space is the collection of variables one controls directly. For a vision system this could include the threshold for a detection to be considered valid. For a planning system, this may include cost function parameters. Like the configuration space, the decision space is the set of parameters under the designer's control, but does not represent the actual goals. The objective space, in contrast, contains the objectives of interest. In control or planning, this corresponds to the task space. In autonomous system design, the objectives are often conflicting criteria representing safety and efficiency. In our work, we illustrate our techniques on a simple 1D decision space, a classifier decision threshold and several 2D objective spaces. One observation from MCDM, which simplifies work in higher dimensional spaces, is that we only need to consider non-dominated alternatives. If a particular set of decision variables leads to worse performance in all objectives, it can be immediately removed. In real systems, there are parameterizations where a change in one variable improves both safety *and* efficiency, but we can focus our attention on the Pareto Frontier, the manifold where we can only improve one objective at the cost of another. The same approach can be applied to multi-dimensional decision spaces by exploring the Pareto frontier that results in the chosen objective spaces. For an example of MCDM ideas applied directly to robotic control problems, consider Ariizumi et al. (2016).

2.5. Making safer decisions

For an autonomous vehicle, the safest course is to stay in place; the most productive is to rush headlong along one's path. This trade-off illustrates a fundamental principal of robotics, that controlling a vehicle requires a careful consideration of what it means to operate safely. There is extensive work on robot *decision making* that considers robustness in the context of safety. Since our focus is on perception, we provide only a short summary of that work. While traditional Markov Decision Processes (MDP) focus on long-term reward maximization, our approach is most related to work on constrained MDPs (Altman, 1999; Dalal et al., 2018) whereby safety-signals (observable sensor data) must be kept bounded within certain constraints for each state. Modifications such as this are necessary because in the context of safety we can no longer rely on the greedy pursuit of reward because what we actually want is to avoid a worst-case state transition with highly negative consequences. This concept has been used in the context of reinforcement learning (RL) to guide policies towards those that satisfy state-wise safety constraints (Dalal et al., 2018; Dulac-Arnold et al., 2021) without the burden of unraveling the unpredictable effects of reward shaping. By minimizing the triggering of these signals, researchers were able to identify policies describing how to respond to state transitions appropriately in the context of unexpected failures. However, in our work we observe that the real-world decision space is often not a complex control policy to avoid unsafe states but rather a decision about the level of constraint that works best in the context of real-world data. In our case as in many others the problem of robust safe control is defined first and foremost not by the optimality of the actions we select but to what extent the system design properties (decision thresholds for pedestrian detection) can impact policy performance in the wake of environmental mutations. By representing robust perception in the context of a ROC curve instead of with success-rate or mean reward (Peng et al., 2018) we show that it is possible to understand robustness in way that can directly inform these system design choices before, during and after their deployment.

3. Motivating application

We demonstrate our approach on a perception module for person detection, used to safeguard a system that controls an autonomous tractor in an orchard environment. Prior work on such a robotic system is described in Moorehead et al. (2012) and Stentz et al. (2002); the NREC-AgPD dataset used for this analysis is introduced in Section 3.2. We expect our approach to generalize across a variety of robotic system applications and datasets.

3.1. Problem definition

The autonomous tractor (shown in Figure 1) is designed to carry out mowing and spraying operations in an orchard environment with minimal supervision from a remote operator. The safeguarding system under evaluation is responsible for ensuring the vehicle comes to a stop if there is a person in its way, followed by asking a remote operator for help. Other modules in the system handle path planning for the vehicle, so the safeguarding system's sole output is a safe speed value that guarantees the vehicle is able to stop in time for any detected people in its path. As a result, the system must not only be able to detect people, but also estimate their location relative to the vehicle's trajectory.

Rephrased in terms of safety invariants, the module's responsibility is to guarantee that the vehicle is never in motion within a specified distance (safety radius) of a person. For simplicity, we define this distance as a path along the ground, between the camera and the person's centroid, neglecting further extents of the person or vehicle extremities. This may make small differences in estimated person-robot distances, but should not affect the gross performance and robustness trends we focus on in this work. A true safety-critical system would require more rigorous treatment, but we considered this sufficient to demonstrate our approach.

Note that in this application the safeguarding system is programmed to use a control radius that is larger than the safety radius. i.e., the control algorithm tries to stay even farther away from people than would constitute a safety violation. This enlarged radius exists because having a goal exactly equal to the safety radius effectively guarantees safety violations from the slightest errors in estimation or control. The difference between them can be viewed as a further safety buffer for the safeguarding system.

Our evaluation builds upon previous work on this application that includes modeling of the vehicle behavior in log playback, along with associated ground truth (desired) behavior (Dima et al., 2011).

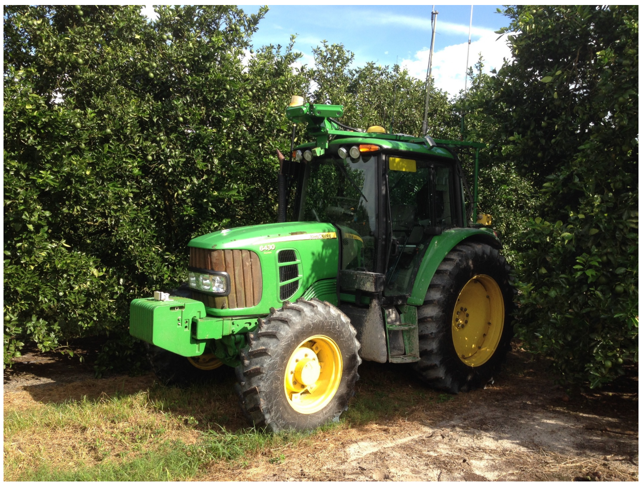


Figure 1. Tractor with stereo camera setup for motivating application.

3.2. The NREC Agricultural Person-Detection Dataset

The NREC Agricultural Person-Detection Dataset (NREC-AgPD dataset) consists of stereo video of people, annotated by 2D bounding boxes, in orange and apple orchards, along with vehicle position data from RTK GPS. It contains a total of 76k labeled person images and 19k sampled person-free images, all with associated stereo. The dataset was released to the public in 2017 along with the publication Pezzementi, Tabor, Hu, et al. (2018). We chose this dataset as an example of a safety critical application of object detection. It includes diverse examples of dangerous situations where a person is in the way of an autonomous vehicle.

4. Method overview

Our general approach to robustness evaluation is to obtain real images and associated ground truth output, mutate the images (discussed in section 5), and monitor how each algorithm's (which we refer to as a System Under Test or SUT described in Section 7.1) output changes under these mutations. A robust SUT's output should ideally not change under mutation; practically, the change should be as small as possible.

We train the SUTs on the NREC-AgPD dataset using the default parameters from their respective publications or source code. The NREC-AgPD dataset's validation set is used to evaluate training convergence, at which point we choose the training iteration with the best validation performance for evaluation. All results presented are computed on the subset of the test set (as per the benchmark Pezzementi, Tabor, Hu, et al. (2018)) that meets the requirements for the analysis.

In our previous work (Pezzementi, Tabor, Yim, et al., 2018), we demonstrated this approach using traditional ROC curves, evaluating the robustness of algorithms in terms of how much the area under the curve changes for inference on mutated images. This technique matches the standard practice for evaluation of detectors in the research literature and has the advantage of evaluating a range of possible system configurations. However, it is difficult to define requirements, based on this metric alone, that represent when a robotic system would be ready for deployment. Its maximal generality also prevents it from distinguishing between error types, some of which may be more or less important to a particular application.

We therefore introduce generalizations of the ROC curve that still capture the essential trade-off between safety and efficiency, but introduce application-specific context that makes them easier to relate directly to system-behavior requirements.

Figure 2 illustrates observations of key shortcomings of basic ROC curves:

- People close to the vehicle are usually more important to detect than those far away.
- Only people in the vehicle path are relevant to the safety requirements.
- Intermittent detections can be acceptable as long as the vehicle is still able to maintain the safety requirement.

We therefore introduce context gradually to incorporate performance metrics that account for these observations:

- 3D locations of people
- The planned vehicle path
- How quickly the vehicle can speed up or stop

The details of the approach are covered in the following sections: Section 5 describes the ways in which we mutate input images to simulate adverse conditions. Section 6 goes over the way in which the context mentioned above is incorporated into our data pipeline to facilitate analysis at several different points. Section 7 describes our input data sources, including both ground truth and the person-detection algorithms being evaluated. Then Section 8 introduces our novel performance/robustness metric, as well as how it can be applied at each context level. This metric operates

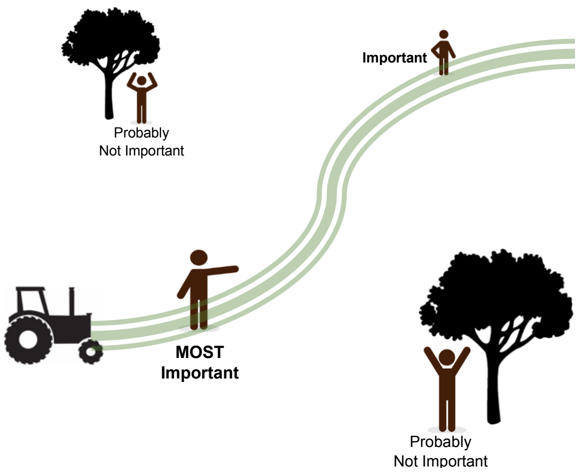


Figure 2. Illustration of the varying importance to detect people in different locations for the sample application, showing how additional context can change the appropriate performance evaluation.

on general, multi-objective trade-off curves that represent the range of performances one can realize by varying system configuration, and it ensures that gains under mutation on one objective can not hide losses on another objective.

5. Mutations

The mutations we use in this work are a subset of those from Pezzementi, Tabor, Yim, et al. (2018), which we describe in detail here due to their centrality to the approach. We refer to generators of image perturbations as well as the perturbation itself as "mutations" with two classes: (1) "Simple" mutations that only require the original monocular image, and (2) "Contextual" mutations that incorporate additional information, such as high-fidelity depth estimates. We demonstrate the usefulness of scene geometry for realistic simulation of two common physical phenomena: haze and defocus. The presence of both stereo images and continuous video in our dataset allows us to use scene flow techniques to produce simultaneous estimates of scene geometry and optic flow.

5.1. Simple mutations

5.1.1. Gaussian blur

Blur the image using a symmetric Gaussian kernel with standard deviations of $\sigma = \{1.5, 2, 2.5, 3.0\}$ pixels. This can be seen as a simple approximation of effects such as defocus or material on the lens.

5.1.2. Alpha blend haze

Alpha blend the image with a uniform color, using blending coefficients of $\alpha = \{0.1, 0.25, 0.5, 0.75\}$. We use a gray color similar to fog, $h = [205, 208, 211]$ in RGB, to provide a simple approximation of haze effects.

5.1.3. Channel drop-out

Simulate channel drop-out by setting the pixel values of various channels to 0. We applied this to the $ch = R, G, B$ channels in RGB space and the $ch = Cb, Cr$ channels in YCbCr space. This can occur in real systems if the individual channels are transmitted in separate communication packets. If one packet is lost, you may still need to operate on the remaining data.

5.2. Contextual mutations

To calculate the additional information required for the contextual mutations, we estimate depth throughout every image in the test set by applying Piecewise Rigid Scene Flow (PRSM) (Vogel et al., 2015), the leading method with a public implementation on the KITTI Scene Flow Benchmark (Menze & Geiger, 2015). We seed the estimation with some prior knowledge of the vehicle hood location (which does not move) and rectification artifacts (considered planes at infinity).[1] We then refine the estimated depths using a bilateral solver (Barron & Poole, 2016) to maintain good alignment between depth and image edges. This provides high-fidelity estimates of depth at every pixel, $\mathbf{D}(x)$, for each image. Performing this on a large dataset is time-consuming, but only needs to be completed once.

5.2.1. Depth-varying haze

We use a uniform haze model that has been widely applied and shown to be effective in haze removal (He et al., 2011), which models haze as an alpha-blend effect whose alpha term increases with distance from the camera:

$$\mathbf{H}(x) = \mathbf{I}(x) \cdot \mathbf{T}(x) + h\,(1 - \mathbf{T}(x)) \tag{1}$$

$$\mathbf{T}(x) = e^{-\beta \mathbf{D}(x)} \tag{2}$$

$\mathbf{H}(x)$ is an image under the effects of haze. $\mathbf{I}(x)$ represents the scene radiance, in our case the original image, and $(\cdot)$ indicates element-wise multiplication. Scene radiance is attenuated exponentially by transmission, $\mathbf{T}(x)$, as depth increases, shifting it toward the color of the haze, h. Before applying Equation 2, we smooth $\mathbf{D}(x)$ with a Gaussian filter ($\sigma = 2$) to soften discontinuity effects. β captures the density of the haze, which can be converted to a visibility distance, u_V, by applying the Koschmieder formula : $u_V = \frac{3.912}{\beta}$. In our experiments, we used a single gray haze color matching that from Section 5.1.2 and density/visibility values shown in Table 1. 100 m is the lowest visibility distance reported by NOAA before rounding to “zero” (“Visibility”, 2018), so that is the most extreme condition we simulate. Sample images of the result are shown in Figure 4.

5.2.2. Defocus

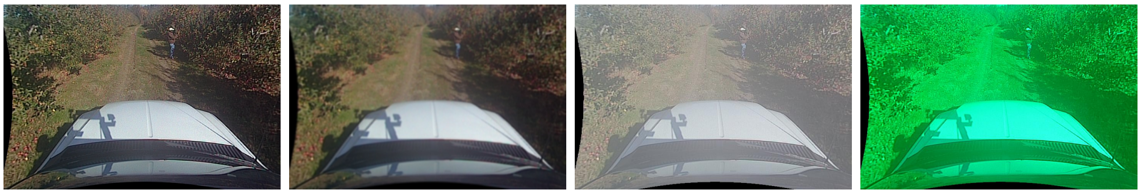
We use a common model of defocus as scattering light by a point-spread-function and distributing it to many pixels in the image. This spread depends on the depth of the scene point each pixel normally images. To mutate an image to exhibit defocus, we first compute the spread for all incoming pixels, then compute the proportion of their color information that is delivered to every other pixel, $g(x, y, \mathbf{D}(y))$. Then we sum over the effects of each input pixel and normalize.

$$\mathbf{F}(x) = \frac{\sum_y I(y) g(x, y, \mathbf{D}(y))}{\sum_y g(x, y, \mathbf{D}(y))} \tag{3}$$

Table 1. Parameters used for haze and defocus mutations. Haze scattering coefficients, β, are shown with resulting visibility distances, u_V. For the defocus mutation, a test was run for every combination of focus distance, u_f, and camera constant, κ, values. Reproduced from Pezzementi, Tabor, Yim, et al. (2018).

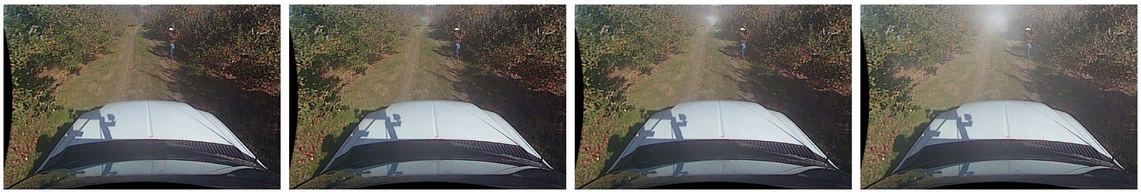
Mutation	Variable	Values		
Haze	β	0.04	0.012	0.004
	u_V	97.8 m	326 m	978 m
	h	RGB (205, 208, 211)		
Defocus	κ	2.0	2.8	
	u_f	1 m	2 m	5 m

[1] PRSM updates available at https://github.com/vogechri/PRSM/pull/2



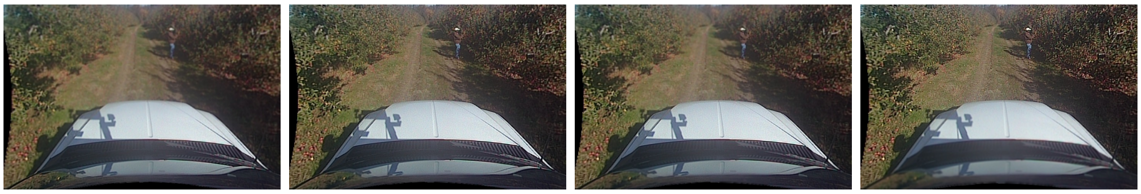
(a) Original Image, **I** **(b)** Gaussian Blur, $\sigma = 2$ **(c)** Alpha Blend, $\alpha = 0.5$ **(d)** Channel Drop-Out, $Y(Cb)Cr$

Figure 3. Simple mutations applied to a sample image from the NREC-AgPD dataset.



$u_V = 3260\,\text{m}$ $u_V = 978\,\text{m}$ $u_V = 326\,\text{m}$ $u_V = 97.8\,\text{m}$

Figure 4. Example images of different magnitudes of artificial haze applied to the same image as in Figure 3a. Choice of visibility distances is described in Section 5.2.1.



$u_f = 1\,\text{m},\ \kappa = 2$ $u_f = 2\,\text{m},\ \kappa = 2$ $u_f = 2\,\text{m},\ \kappa = 3.6$ $u_f = 5\,\text{m},\ \kappa = 3.6$

Figure 5. Example images of artificial defocus on the same image as Figure 3a. These cover some samples from our two controllable parameters for defocus simulation. u_f is the distance at which the camera is focused. κ is the camera constant. Reproduced from Pezzementi, Tabor, Yim, et al. (2018).

We assume that incoming light is scattered by a 2-dimensional Gaussian point spread function, $g(x, y, \mathbf{D}(y))$. The mean is the incoming image location, y, and the function is evaluated at the output image location, x. The standard deviation is the blur radius (Arroyo, 2013), $\rho(y)$, computed by comparing the focus distance u_f to the depth, $\mathbf{D}(y)$, of the scene at that image point y and scaling by a camera constant, κ:

$$g(x, y, \mathbf{D}(y)) = N_2(x, \Sigma(\mathbf{D}(y))) \tag{4}$$

$$\Sigma(\mathbf{D}(y)) = \mathrm{diag}(\rho(\mathbf{D}(y))^2) \tag{5}$$

$$\rho(\mathbf{D}(y)) = \kappa \frac{|\mathbf{D}(y) - u_f|}{\mathbf{D}(y) \cdot u_f} \tag{6}$$

Choosing a subset of focus parameters from Pezzementi, Tabor, Yim, et al. (2018), we evaluated the effects of two values of κ for each of three focal distances. These values are summarized in Table 1. Mutated image examples are shown in Figure 5.

5.3. Mild and severe groupings

In the robustness evaluation, we divide the mutations above into "mild" and "severe" groups, based on the level of degradation that results on the image.

The mutations in the severe group are

- Gaussian blur with $\sigma >= 2.5$
- Alpha blend with $\alpha >= 0.5$
- Haze with $u_V = 97.8$
- Defocus with u_f 1 and (κ 2; κ 3.6)
- All channel dropout

and all others are in the mild group.

6. Application implementation

Figure 6 illustrates the way we break down the operations performed by a robotic system, facilitating analysis at different levels of modularity, with corresponding levels of application context.

6.1. Roboticizer

The first refinement we perform to add context to the data is to transform detections from the image frame to a robot frame common with the ground truth, by converting the 2D bounding box coordinates into 3D world locations. This is necessary as our SUTs do not estimate 3D locations directly (SUTs that produce 3D detections could of course skip this step). This allows for reasoning about detections in terms of 3D distance, which is more easily tied to robot behavior. We also standardize strengths of detections across all SUTs to a common reference frame.

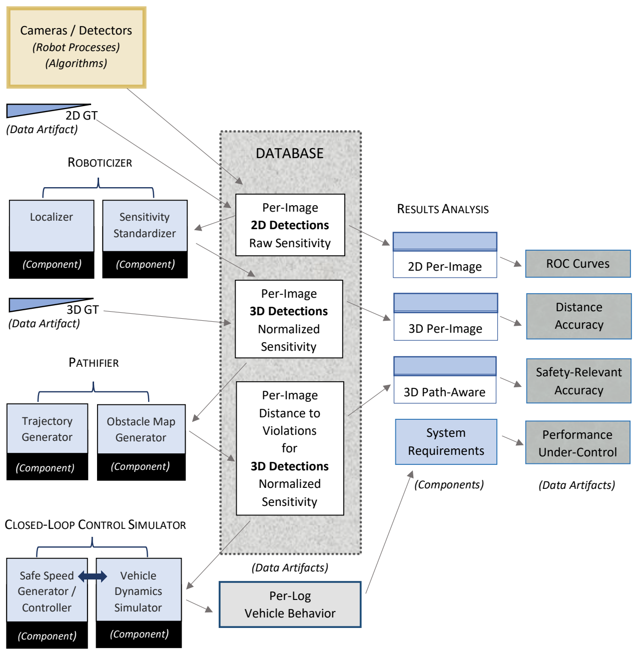


Figure 6. Overview of our process for analyzing person detection performance at different levels of application context: The major robot process components are described in the remainder of Section 6. Intermediate results are stored in a database, facilitating analysis at different levels, explained in Section 8.

6.1.1. Localizer

To convert detections from 2D to 3D, we apply a heuristic that reuses the high-fidelity depth estimates described in Section 5.2 for each image. It would be equally possible to use a real-time method instead though. To perform the localization we take the median depth of each row, then the median of the result (approximating overall median) to estimate the depth of each bounding box. This depth is associated with the box centroid to provide a point position for safety invariate violation calculations.

Note that due to the expense of computing high-quality stereo disparity, we use the depth maps from the un-mutated images for all localization. It is therefore possible for the mutations to result in larger performance drops than we observed by degrading localization accuracy as well.

6.1.2. Sensitivity Standardizer

We also perform standardization of sensitivities across SUTs at this stage. Without standardization, detection strengths for each SUT can have wildly different ranges.[2] In order to create a level basis of comparison, we remap detection strengths to be in terms of each SUT's predicted false positive rate, based on a lookup table computed from the non-mutated validation set—called "Standardized Sensitivity". This metric provides an important reference point to measure the change in false positive rate (with associated threshold) when image mutations are applied. It also provides a convenient space for estimating viable sensitivity ranges for the application, with which many vision researchers have familiarity from one axis of the standard ROC curve for detection. False positive rates above one per image are unlikely to be feasible at the system level for this application; the lower end is less bounded, but rates below 10^{-4} are unlikely to have acceptable safety for any of the SUTs evaluated. We do not restrict the range of false positive rates realized by an SUT though; if a detector has a parameterization that can achieve extremely high false positive rates, we consider it valid.

6.2. Pathifier

In the second stage, we add the context of the planned path of the vehicle. The "Pathifier" module then checks each detection to determine if it lies within the control radius distance of the path forward. The process is illustrated in Figure 7. Adding this information about vehicle motion also makes temporal consistency easier to incorporate.

6.2.1. Trajectory Generator

The Trajectory Generator uses the position information captured within the NREC-AgPD dataset to establish the vehicle's path in each log. Later steps in simulation will lock the vehicle motion to along this path, but allow changes in the speed at which it is traversed. Since every log with a person ends with the vehicle coming to a stop safely, we extend the path forward 10 m to allow modeling of the possibility that the vehicle would continue. We do this by fixing the vehicle's orientation and translating forward in a direction given by a smoothed estimate of the vehicle's current motion at the end of the log[3]. Although we of course can not evaluate what the system would perceive at these extrapolated positions, we can use them for establishing where detections lie with respect to the robot's path for positions all the way up to the end of the original log. This in turn allows for finer analysis of detections' relevance to safety requirements.

[2] Different frameworks and implementations use various conventions for representing detection strength, with ranges of $[-\infty, \infty]$, $[-1, 1]$, $[0, 1]$, etc. Even the detections from two networks using a $[0, 1]$ range are typically not comparable, since these detections do not generally represent true probabilities; this can sometimes be resolved with some post-processing techniques, such as temperature/Platt scaling (Guo et al., 2017), but probabilities turn out not to be the most convenient representation for further system-relevant analyses.

[3] We fit a cubic spline to parameterize motion along each axis of the path points and then take the derivatives at the final point as estimate of current direction. This gives values close to the true direction except during strong turns at the ends of rows; those situations are a small part of the dataset though, and the camera field of view is not wide enough to fully cover them. A real system would use additional sensors to cover them, which is beyond the scope of this analysis.

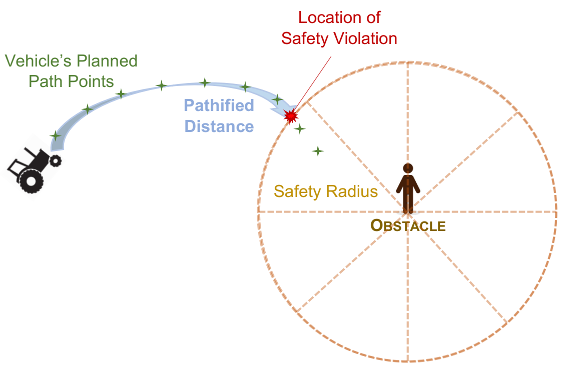


Figure 7. Pathifier process: Obstacles are expanded by the control radius and checked for intersection with the vehicle path. The location at which a safety violation would occur is computed, and the "pathified" distance is the length of the path to this point.

6.2.2. The Obstacle Map Generator

The Obstacle Map Generator compares the locations of all detections in a given image with the associated path forward to determine if any lie within the control radius. For any that do, it finds the intersection of the path with a cylinder around the detection with radius given by the control radius. It then computes the distance along the path to that point, which we call the "pathified" detection distance.

Assumptions:

- The path from per-image pose data, before interpolation, is densely-sampled enough that grazing[4] the obstacle's safety radius is not significant. i.e., the safety radius can be enlarged by an amount, ϵ, to account for this, but ϵ is small enough to be negligible.
- If the position of the most recent image frame is within a detection's safety radius, then so is the current robot position. Our vehicle's viewing geometry should guarantee the next image frame to also be in the safety radius, and therefore any point along the segment between them to be in violation as well. If this assumption is violated though, it results in an overly-cautious system, not a compromise on safety; there is a chance of a loss of efficiency in this rare condition.

6.3. Closed Loop Control

Our final level of application context is simulating closed-loop control using log playback, as in Dima et al. (2011). This is the level of simulation that most accurately models the robot's behavior and provides requirements-level metrics for fielding a system. To achieve this behavior, it requires modeling the robot's dynamics as well as defining and implementing a particular controller to govern its behavior. This integration of the perception system with the the downstream controls system ultimately captures more accurately how these scenarios affect the whole system behavior.

We use a simple vehicle model with a top speed and linear acceleration with separate maximum and minimum values. We set max speed, $s_{\max}$, to $3\,\mathrm{m/s}$, matching the speeds used during the data collection when no obstacles were present. For minimum acceleration, we used the legally required braking distance for vehicles of the size used during data collection (Transportation, 2019), with an adjustment for latency to start braking, giving $a_{\min} = -15\,\mathrm{ft/s^2} = -4.572\,\mathrm{m/s^2}$. We used $a_{\max} = 5\,\mathrm{ft/s^2} = 1.524\,\mathrm{m/s^2}$ for max acceleration based on the vehicle and terrain. We chose a simulation update rate of $50\,\mathrm{ms}$, the smallest increment that still yielded significant differences.

[4] We refer to grazing as the case where two consecutive poses are outside the radius, but line between them enters the radius.

The Safe Speed Generator is responsible for finding speeds the vehicle may travel without violating the control radius. The operations of Pathifier simplify it to a one-dimensional control problem. The generator finds the positive detection with the smallest pathified distance, d_p, and determines the fastest speed that still allows the vehicle to stop (with max braking[5]) before traversing that distance:

$$s_{\text{safe}} = \min(\sqrt{-2a_{\min}d_p}, s_{\max}) \tag{7}$$

For any detections where this can not be achieved (e.g., a sudden detection that is already within the control radius), safe speed is set to zero.

When both safe speed and vehicle speed reach zero, a stop is recorded. If running on positive logs, this is the end of the run, but on negative logs, we allow the vehicle to start again. This simulates an operator intervention where all obstacles from the most recent image have been cleared (which should not happen on positive logs, where there actually is a legitimate obstacle in view). The vehicle can then accelerate again from zero speed until encountering the next image, which may or may not have new obstacle detections. This ensures that all runs will evaluate the full length of the negative logs, regardless of the frequency of false detections.

The Safe Speed Controller takes as input the current vehicle state and a target safe speed from the last captured image and simply accelerates/decelerates at the max rate until achieving the target speed.

We record the vehicle state at each time step of the simulation, which allows for a wide range of analyses, as described in Section 8.2.3.

Assumptions:

- Changes in the time between images when simulating a different vehicle speed are not significant.
- The camera position is a sufficient representation of vehicle position.
- Consistent yaw rates from the camera perspective are realistic when extrapolating vehicle paths.
- At the rates we are deviating from the original speed, the effects of rotation on vehicle speed are negligible.

7. Data sources

As illustrated in Figure 6, input detection data for the pipeline can come from different sources, either the detectors being evaluated or our two different forms of ground truth.

7.1. Systems Under Test (SUTs)

The full set of SUTs evaluated is shown in Table 2. It covers a variety of architectures and libraries (with associated training procedures) that display a range of performance and robustness characteristics, illustrating how our approach can distinguish between them.

Object detectors can be divided into two categories: one-stage and two-stage. One-stage detectors directly predict bounding boxes and classifications in one forward pass. Two-stage detectors are composed of a first stage that proposes regions of interest and a second stage that classifies and refines these proposed regions. The former are more naturally suited to the limited computation of embedded perception.

The one-stage detectors tested in this work include:

- *The Single Shot Detector (SSD) (Liu et al., 2016) meta-architecture with Inception V2*, the ILSVRC 2014 classification and detection winner, as a feature extractor. Inception V2 utilizes modules that concatenate efficient decomposed filters (Szegedy et al., 2016).

[5] This results in an aggressive controller that drives the robot at max speed until about 1 m from a safety violation. In practice, one would probably want a controller that brakes more softly and therefore reacts from farther away; in the limiting conditions for the worst-case safety analysis though, it reduces to the aggressive controller described.

Table 2. List of SUTs evaluated.

SUT	Base Library	Reference(s)
MS-CNN	Caffe	Cai et al. (2016)
SSD w/ MobileNet	TensorFlow	Howard et al. (2017), Huang et al. (2017), and Liu et al. (2016)
SSD w/ Inception	TensorFlow	Huang et al. (2017), Liu et al. (2016), and Szegedy et al. (2016)
R-FCN w/ ResNet-101	TensorFlow	Dai et al. (2016), He et al. (2016), and Huang et al. (2017)
Faster R-CNN w/ ResNet-101	TensorFlow	He et al. (2016), Huang et al. (2017), and Ren et al. (2017)
Faster R-CNN w/ Inception ResNet v2	TensorFlow	Huang et al. (2017), Ren et al. (2017), and Szegedy et al. (2017)
Deformable R-FCN	MXNet	Dai et al. (2016), Dai et al. (2017)
Deformable Faster R-CNN	MXNet	Dai et al. (2017) and Ren et al. (2017)
YOLOv2	Darknet	Redmon and Farhadi (2017)
YOLOv3	Darknet	Redmon and Farhadi (2018)
RetinaNet	PyTorch	Girshick et al. (2018) and Lin et al. (2017)
Faster R-CNN w/ Inception ResNet FPN	PyTorch	Girshick et al. (2018) and Li et al. (2018)

- *The Single Shot Detector (SSD) meta-architecture with MobileNet* as a feature extractor. MobileNet is optimized for computational efficiency with filters that are further decomposed (Howard et al., 2017).
- *YOLO v2* which improves on the original YOLO (a variation on SSD run on the Darknet framework) by adapting anchor boxes, utilizing multi-scale features, batch normalization, and a number of improvements to training with different resolutions and tasks (Redmon & Farhadi, 2017).
- *YOLO v3* improves upon YOLO v2 by separating predictions of objectness and classes, potentially solving a similar issue that RetinaNet's Focal Loss addresses. Also, the detector adds residual blocks, high level semantics across scales (similar to Feature Pyramid Networks below), and multi-scale predictions (Redmon & Farhadi, 2018).
- *RetinaNet* is a one-stage detector on the Detectron framework (Girshick et al., 2018) with a *Feature Pyramid Network (FPN)* and focal loss (Lin et al., 2017). FPN adds higher level semantic information at all spatial scales. By passing the later layers information back to earlier higher resolution scales before predictions, performance on smaller objects should be improved significantly. Focal loss allows the network to gradually focus on the hardest examples throughout training. This fixes a major problem with one-stage detectors: class imbalance of foreground and background across all potential locations.

The two-stage detectors tested in this work include:

- *Faster R-CNN meta-architecture with Resnet-101.* Faster R-CNN uses a region-proposal network to generate proposed object regions and a box classifier to refine each of these proposals. The ResNet-101 feature extractor, which won the ILSVRC 2015 and COCO 2015 classification and detection, uses residual connections to train very deep networks (He et al., 2016).
- *Faster R-CNN meta-architecture with Inception Resnet*, another feature extractor that enhances the Inception modules with residual connections and *à trous* convolutions (Szegedy et al., 2017). We include both standard and FPN implementations.
- *MS-CNN*, a modified Faster R-CNN that generates region proposals with multiple scales through deconvolution of the feature maps, with VGGNet (Cai et al., 2016).
- *R-FCN meta-architecture with Resnet-101.* The R-FCN meta-architecture shares most of the classifier stage feature computation across the proposals by cropping in the later steps of the second stage, significantly reducing computation (Dai et al., 2016).

- *Deformable Convnets* enable learned free-form deformation of the sampling grid (Dai et al., 2017) and ROI pooling. The flexibility of the boundaries helps the model localize non-rigid objects.

7.2. Ground truth

Key to detection in computer vision are bounding box labels of objects of interest. The NREC-AgPD dataset used in this work is annotated with standard 2D bounding boxes, which enables standard detector ROCs. For more contextual metrics, 3D bounding box information is required. We applied two different techniques for generating ground truth 3D locations.

7.2.1. Human annotation

One source of 3D ground truth we use to evaluate detectors comes from human annotations. We labeled a subset of the NREC-AgPD dataset with 3D point locations for each labeled person. In order to label this stereo dataset efficiently and accurately, we developed a process where an annotator only needed to label a single point in the left image and its corresponding point in the right image for every stereo frame. These points were then used to determine the depth of the person using stereo triangulation. This depth was applied to the center of the bounding box to generate its final 3D position. In order to ensure consistency (and efficiency) in labeling these point correspondences, we added a few simple enhancements:

- The epipolar constraint was enforced, causing the point in the right image to jump to the epipolar line.
- A point tracker from OpenCV was integrated, to try to propagate the correspondence through the log.
- A graph of depths over the log was displayed. Discontinuities in this graph provided guidance on which correspondences to improve, whether manually or automatically added.

7.2.2. Automatic localization

The other source we use for 3D ground truth is to apply the Localizer module, described in Section 6.1, to the original 2D bounding boxes. This means that the ground truth is going through the same process as the detections generated by a SUT. Failures due to the Localizer module are not counted against the system unnecessarily. For example, if there are holes in the depth maps used by the Localizer, those holes cause both the localized 2D-to-3D ground truth results and the localized SUT results to fail in the same way.

8. Performance evaluation

Prior work has focused on generating trade-off (usually ROC or precision-recall) curves using area under the curve (mAP or average detection rate) with and without mutation to measure robustness. We base our analysis on related areas under trade-off curves, allowing evaluation of algorithms across a range of configurations, so that one can defer final configuration tuning to a later stage of the design process. We introduce a variety of ways to generate trade-off curves: we begin with the standard ROC curve that predominates in the previous literature (ROC_A, in this section), then show a minor modification of it that is more consistent with those that follow (ROC_B, in Section 8.1), then introduce new trade-off curves that incorporate more system-level context (in Section 8.2). Finally, Section 8.3 goes over new metrics for comparing baseline to mutated curves, which can be applied to any of these curves.

The trade-off curves in this work each measure safety on one axis and efficiency on the other and sweep across module configurations (in our case detection thresholds) to realize points along a curve. Safety is some measure of how low we can drive the probability of an injury to a person that encounters the system. Efficiency is a measure of how much of the total operation of the system is spent doing useful work (instead of slowing or stopping needlessly). This definition of efficiency is similar to that of Hollnagel (2009) from the risk analysis literature, but we choose not to adopt their opposite trade-off of thoroughness, since that applies to safety focused effort performed before the system is deployed. When we are discussing these trade-off curves, we will also refer to “performance”, which consists of overall behavior combining safety and efficiency, usually measured by area under the curve.

A standard ROC, denoted ROC_A, is a valid and very common type of trade-off curve to use for such analysis. This is the metric used in the official benchmark for the dataset. Typically, the y-axis shows detection rate (recall), and the x-axis shows the average false detection rate per image, given by a ratio of total false positives to total images:

$$FPR_A = \frac{\sum_{i=1}^{nI} nFP_i}{nI} \tag{8}$$

In that case one can use recall directly for the safety axis, and one can obtain an efficiency value by inverting the false detection rate:

$$E_{ROC_A} = 1 - \min(FPR_A / FPR_{\max}, 1) \tag{9}$$

Since the false positive rate is unbounded, we impose a ceiling of $FPR_{\max}$, a rate beyond which the resulting system efficiency is impractical; we use $FPR_{\max} = 0.1$ in our implementation.

8.1. ROC formulation

We introduce our traditional ROC variant, ROC_B, consisting of two modifications, for consistency with the approach in the other trade-off curves that follow: we only measure false detections on images from negative logs, and we only keep track of the strongest false detection per image.

We focus on negative logs to evaluate the effects of false detections (performance degradation), since these more closely resemble typical operating conditions, where optimal behavior is easy to model and all detections are known to be spurious. False detections also occur in positive logs of course, but it is more difficult to disentangle their effects from those of true (possibly mis-localized) detections in an automated way. That being said, the positive logs are used in this method to evaluate safety in terms of ground truth locations. The other metrics that follow also use this approach of evaluating safety on positive logs and efficiency on negative logs.

Switching to evaluating a single false positive per image means a change in the meaning of false positive rate: instead of representing expected number of false detections per image, it becomes the probability of any false detection in an image:

$$FPR_B = \frac{\sum_{i=1}^{nI} \mathbf{1}(nFP_i)}{nI} \tag{10}$$

where $\mathbf{1}(x)$ is a step function. It also moves us closer to the more contextual trade-off curves, where the effect of the strongest detection on efficiency of a particular parameterization tends to dominate weaker ones’.

We then compute efficiency in the same way as 9, but with the modified way of measuring false positive rate:

$$E_{ROC_B} = 1 - \min(FPR_B / FPR_{\max}, 1) \tag{11}$$

Most performance metrics in the following section use the same approach to compute E, just with different ways of measuring the false positive rate. We include results for both ROC_A and ROC_B to better understand the effects of context that later trade-off curves introduce, as well as to separate them from the modifications made between ROC_A and ROC_B.

8.2. Analysis at different context levels

Incorporating more context will bring the analysis closer to real-world levels, but requires more information about the system to be simulated or captured, along with associated development cost. Analyzing robustness at intermediate context levels helps with understanding trade-off between these factors.

8.2.1. Localizer

After localizing detections to 3D locations, we can categorize detections in terms of distance to the vehicle, which often have a different cost. In addition, we can incorporate distance into a 3D localization metric.

Our sample metric at this stage marks detections as being correct if they are within 1 meter of the location marked in ground truth, instead of using IoU as in the traditional ROC. It also establishes a range of distances from 0 to 10 meters to focus on. Safety is evaluated only on images where the ground truth location falls within that range, and productivity is evaluated only on detections (the strongest per image) within that range. This removes far-away people from the evaluation, since they will not affect system behavior in this application.

8.2.2. Pathifier

After running data through Pathifier, it is much easier to evaluate whether an image contains ground truth or detections that are critical to the behavior of the system (they intersect with the robot's path), and that level of criticality is easier for engineers to map back to system requirements.

We start to incorporate temporal information at this stage. To focus on the detector and avoid analyzing the large space of tracking solutions (though this may be an appropriate direction for some systems), we distill the salient detector properties into this metric. Temporal consistency is analyzed, depending on how far ahead a person/detection is and therefore how urgent it is to react to. We establish two bins of "pathified" distance to a safety violation: strict at a distance of $[0 - 2)$ meters and lenient at a distance of $[2 - 5)$ meters. The strict bin is based on the approximate stopping distance of the vehicle when running at full speed, so missed detections or false detections there are likely to lead to safety violations or hard braking respectively. The latter bin corresponds to situations where there is a bit more time to react, but one would want the robot to begin slowing down for a smooth stop, and false detections could lead to erratic braking. As above, we evaluate safety in terms of what ground truth locations in positive log images fall within these bins and efficiency in terms of false positives on negative log images within these bins.

In the strict bin, the system must get everything correct. Any missed detections count against the recall rate, and any detections on negative logs count as false positives, again based on the strongest detection per image. Detections on positive images are considered correct from a safety perspective as long as they have pathified distance less than or equal to the ground truth. i.e., the system would respond to them at least as conservatively as the ground truth requires.[6]

In the lenient bin, the system is allowed more leeway since there is more time available to react, thus only consecutive mistakes are counted. i.e., a miss is only counted for a frame in this bin if there was also a miss in the previous frame, and detections on negative images are only counted as false positives if there was also a detection in that range in the previous frame.[7] The results from the two bins are fused into a single evaluation of sensitivity for a match or false positive per image and aggregated.

[6] This does allow for "lucky" false detections that would lead to desirable stops to make the system be considered more safe. This effect is legitimate though, and it is assumed that these should also manifest on negative logs and result in a trade-off in efficiency.

[7] Note that the consecutive false detections need not have any association between them; it is enough that two occur in a row, and it is counted as the strength of the weaker of the two.

The end result is a metric that focuses evaluation on the situations most critical to system behavior, but that avoids the complexities and many of the explicit choices required for the closed loop control approach that follows.

8.2.3. Closed Loop Control

We evaluate each SUT at 100 Standardized Sensitivity levels. These range from 10^{-4} to 1 expected false positives per image, to produce points on the trade-off curves expected to be in the relevant range. Note that this corresponds to particular sensitivity values for the SUT which result in those false positive rates under nominal conditions, but they may produce entirely different false positive rates under mutations.

Both safety and efficiency are measured by comparing behavior against a ground truth system, which uses the same control logic, but operates on ground truth data for person locations. Safety is measured as the portion of logs in which the SUT comes to a stop no later than the ground truth system. Efficiency, E is measured in terms of the relative time for the system to traverse each log under control, t_{SUT}, compared to the ground truth's traversal time, t_{GT}:

$$P = \frac{\sum t_{\mathrm{GT}}}{\sum t_{\mathrm{SUT}} + \nu n S_{\mathrm{SUT}}} \tag{12}$$

nS_{SUT} is the number of times an SUT comes to a stop in a log, and ν is an associated cost for stops. Because of the restart logic described in Section 6.3, t_{SUT} evaluates the same path length in all runs, so an over-sensitive system could produce many stops. In some applications, stops require intervention that may be very expensive, leading to a high value for ν. In our experiments, we set ν to zero, but starting again from zero speed still has impact on t_{SUT} that lowers E.

Many other metrics would be possible for either axis, depending on the specific priorities of an application. For instance, efficiency might depend more on the count of full stops (requiring human intervention) than speed, or safety might differentiate between classes of stops at different distances to a person.

8.3. Robustness metrics

One common way to decide if a system is robust is to consider how its performance decays under mutation. In our previous work (Pezzementi, Tabor, Yim, et al., 2018), we compared the area under a detection trade-off curve, such as ROC or Precision-Recall, before and after mutation. This approach is common in the literature (von Bernuth et al., 2019; Michaelis et al., 2020; Pezzementi, Tabor, Yim, et al., 2018; Volk et al., 2019; Zhang & Wang, 2019), as discussed in Section 2.1. Notably, in Figure 9, we see a trade-off curve where this method provides a good estimate of expected system performance: the area for the SUT on the baseline data, in solid green, was 0.734, but performing a mutation results in a trade-off curve, in dashed magenta, with an area of 0.678. However, comparing these areas can often hide important changes in system behavior. For instance, Figures 10 and Figure 8 show how the trade-off curve can change without significantly affecting the area. In Figure 11 (and to a lesser extent Figure 10), we see that the area under the curve can even increase.

8.3.1. Worst case robustness

Instead of comparing the area under the curve for the baseline and mutated test set, we propose looking at the area of Robustness ROC, the worst case detector performance for that mutation. In the worst case, we consider whichever condition leads to the worst performance for a particular detector and threshold, for both safety and efficiency values. For two input trade-off curves, $\mathcal{R}_X$ (typically Baseline and mutated) with associated safety, $S(X, \tau)$, and efficiency, $E(X, \tau)$, at each sensitivity, τ, the Robustness ROC is given by

$$S(\mathit{RobustnessROC}(\mathcal{R}_A, \mathcal{R}_B), \tau) = \min(S(\mathcal{R}_A, \tau), S(\mathcal{R}_B, \tau)) \tag{13}$$

$$E(\mathit{RobustnessROC}(\mathcal{R}_A, \mathcal{R}_B), \tau) = \min(E(\mathcal{R}_A, \tau), E(\mathcal{R}_B, \tau)) \tag{14}$$

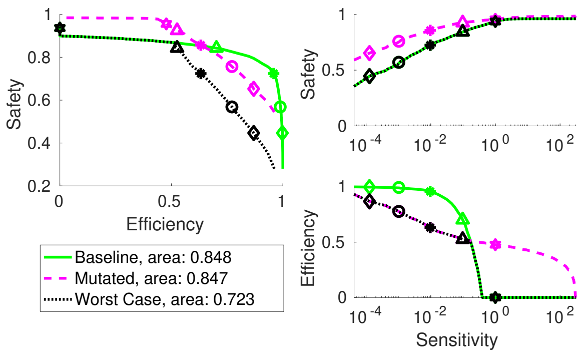


Figure 8. Example trade-off curves (*MS-CNN on channel dropout, G from RGB*) using the Pathifier evaluation. Note how our new worst case detector curve, in dotted black, uses the efficiency and safety metrics from the baseline and mutated detectors, depending on the sensitivity. Symbols along each curve correspond to particular values of sensitivity, illustrating how the safety and efficiency values are derived from this underlying parameter. Sensitivity range depends on the detector; in this case MS-CNN had a set of parameters producing hundreds of false positives per image.

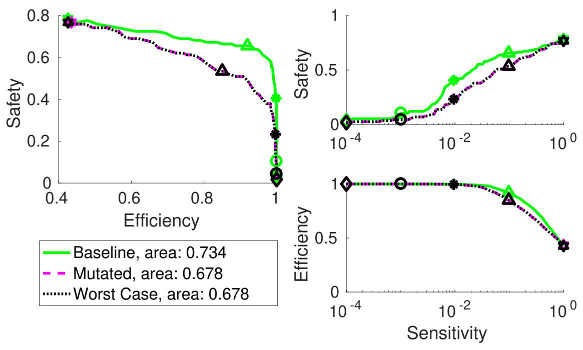


Figure 9. Example trade-off curves (*SSD with MobileNet on Gaussian blur* $\sigma = 1.0$) using the Closed Loop Control evaluation. Note that the area for the mutated detector is equal to our new worst case detector area.

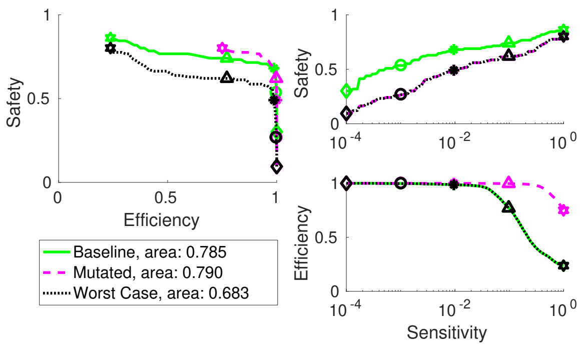


Figure 10. Example trade-off curves (*Faster R-CNN with ResNet-101 on channel dropout, Cb from YCbCr*) using the Closed Loop Control evaluation. Note that the area for the baseline detector is nearly equal to the area of the mutated detector but our new worst case detector area detects a change in robustness.

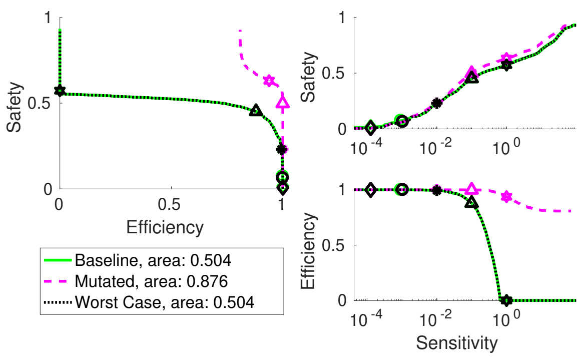


Figure 11. Example trade-off curves (*SSD with Inception on Gaussian blur* $\sigma = 1.0$) using the Roboticizer evaluation. Note that the area for the mutated detector is much greater than the area of the baseline detector, but our new worst case detector does not unfairly credit this improved performance. Sensitivity range depends on the detector, in this case *SSD* had a set of parameters with tens of false positives per image.

Algorithm 1. Robustness ROC algorithm. Shows algorithm for getting worst case ROC, or Robustness ROC from two independent ROCs with sensitivity values that do not correspond. *prev* and *next* are functions that return the nearest Sensitivity in a ROC for the input Sensitivity, in the forward and reverse direction, respectively. When the Sensitivity exists in the ROC, both simply return that value. In absence of a Sensitivity value lower or greater than the query value, respectively, they return the lowest possible value for safety or efficiency.

```
Result: Robustness ROC, $TO_{\text{worst}}$
Input:
$TO_A \subset \{G1_A, G2_A, S_A\}$.
$TO_B \subset \{G1_B, G2_B, S_B\}$.
for $S$ in $S_A \cup S_B$ do
    $S(RobustnessROC(\mathcal{R}_A, \mathcal{R}_B), \tau) = \min \begin{pmatrix} S(\mathcal{R}_A, prev_A(\tau)), & S(\mathcal{R}_B, prev_B(\tau)) \\ S(\mathcal{R}_A, next_A(\tau)), & S(\mathcal{R}_B, next_B(\tau))) \end{pmatrix}$
    $E(RobustnessROC(\mathcal{R}_A, \mathcal{R}_B), \tau) = \min \begin{pmatrix} E(\mathcal{R}_A, prev_A(\tau)), & E(\mathcal{R}_B, prev_B(\tau)) \\ E(\mathcal{R}_A, next_A(\tau)), & E(\mathcal{R}_B, next_B(\tau)) \end{pmatrix}$
end
```

Robustness ROC can then be viewed as a pessimistic lower bound on the performance across these two conditions, but with the important guarantee that gains in one objective do not offset losses in another; system performance must be maintained in all objectives (or show pareto improvements) to avoid the area under the Robustness ROC decreasing.

Consider Figure 8, on the left side of the figure, we show the relationship between each of our objectives, safety and efficiency, and the sensitivity of the detector. The black dashed line shows the worst case choice for each sensitivity, on each of these objectives. Note how on efficiency, we use the performance of the baseline detector for some regions of the curve, and the performance of the mutated detector in others, depending on which is worse. On the left side, we see how these can be combined to create a new curve, Robustness ROC. For illustration, we use markers on the curves to show corresponding sensitivity values. Returning to Figure 11, we see that the area of this new trade-off curve is equal to the original area. In fact, the worst case area will always be between 0 and the baseline detector area. If we consider robustness to be the ratio of these areas, it is guaranteed to be between 0 and 1.

Depending on your choice of evaluation, it is very likely that the sensitivities will not have the same values for the baseline and mutated trade-off curves. In this case, you will need to compare the safety and efficiency of the closest available sensitivities. This generalization is shown in Algorithm 1.

We avoid more standard interpolation of the data, since it introduces new performance values that may not be realizable.[8] For the best accuracy, it is therefore important to have data that cover the range of S-E trade-off values well; otherwise these interpolations and (particularly) extrapolations can have a large impact on area under the curve. Nonetheless, in the absence of such data, the metrics will not produce over-confident, unsafe conclusions.

8.3.2. Any mutation robustness

This same Robustness ROC process of taking the minimum in objective space of each value in the decision space to create a new curve can also be applied repeatedly, in order to build an understanding of the worst-case trade-off for combinations of mutations. This worst case curve then has area less than or equal to the area of any of the baseline or mutated curves. Since the process consists of taking minima, applying it iteratively is equivalent to taking the minimum over the entire set.

We evaluate two versions of this metric, with different sets of possible mutations: "Any" includes all mutations computed in the set, from both the mild and severe groupings; "AnyMild" includes only the mild group.

An alternative method for combining performance from multiple mutations was used in Michaelis et al. (2020), where the areas of each trade-off curve are averaged over all mutations and all severity levels. Conceptually, instead of considering the worst-case mutation for a detector and threshold combination, this is considering a world where each mutation is equally likely to occur. In the case where one knows precisely how common each condition will be in the field, one can take an alternative approach: For every sensitivity value, one can compute average false positive and false negative rates weighted by their probability. Then the area under the curve formed by these points provides an estimate of overall average performance, which could be an informative metric as well; however, we considered the requirement to know the distribution of conditions overly burdensome. In contrast, the Robustness ROC provides a useful lower bound on performance without adding additional parameters.

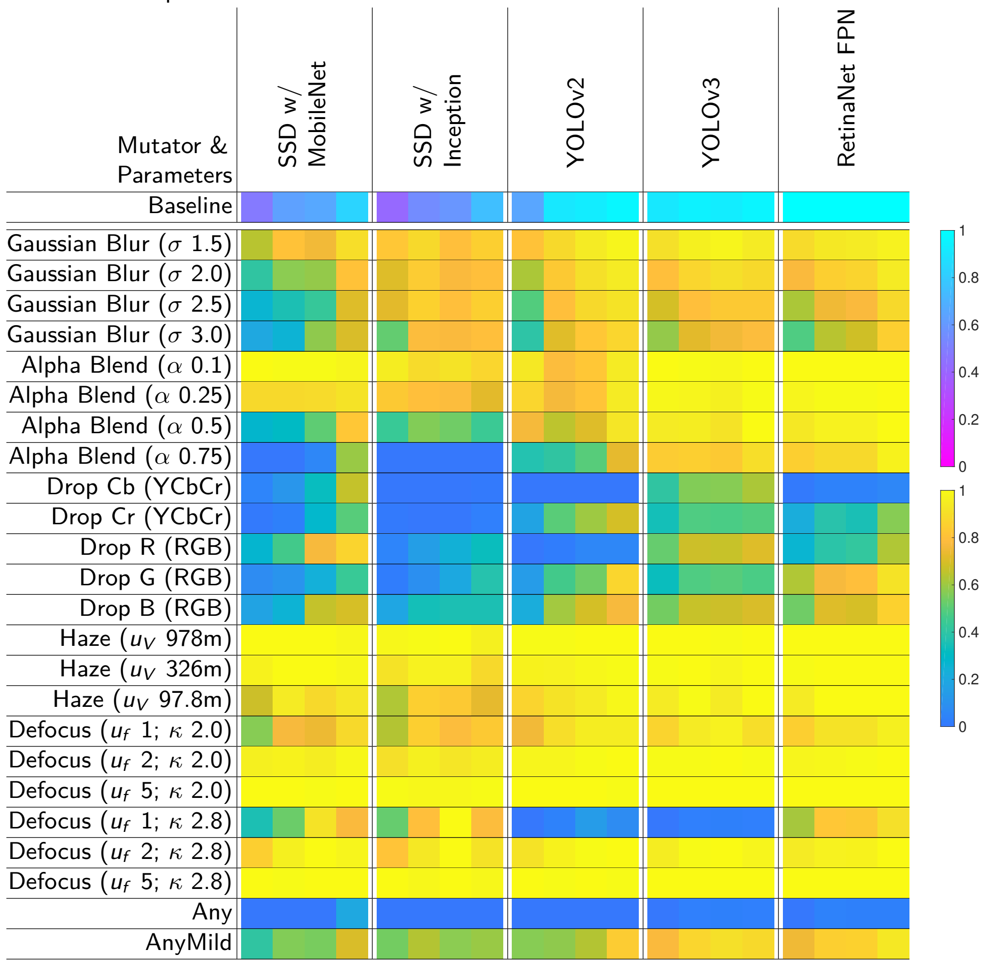
9. Results

Table 5 shows the performance of all Systems Under Test under all mutations, using a traditional ROC performance metric and a worst-case robustness metric. The first row shows baseline performance on unmodified images, while subsequent rows show the mutated performance as evaluated by Robustness ROC, as described in Section 8.3.1. Numbers in each cell show the area under the worst-case curve, while colors show the robustness score, the ratio of that area to the baseline (equivalent to rPC from Michaelis et al. (2020)). This is similar to the results table of Pezzementi, Tabor, Yim, et al. (2018), except using the new Robustness ROC metric. One can see general robustness trends by perusing the colors of columns; robust Systems Under Test feature predominantly warm colors throughout, while cool colors highlight robustness problems. Note that all of these colors are relative to the baseline performance though, so an SUT with a high baseline score and severe degradations under mutation may still outperform a more robust system with a lower baseline score.

Tables 6–10 show the equivalent values for the remaining performance metrics. Tables 5 and 6 show that values for ROC_B do not change substantially from ROC_A. Performances go up slightly, with rankings and robustness trends largely maintained. Tables 7 and 8 show that performance trends also do not change significantly when switching between the 2D and 3D versions of ground truth;

[8] Consider, for example, the extreme case of a detector with detections for every person, but which ranks all candidate detections in exactly the wrong order: all false positives have higher strength than true detections. We may have only tested enough to know the trivial extreme points, where it can (a) achieve perfect safety with zero efficiency or (b) achieve zero safety with perfect efficiency. Standard interpolation would assume a performance with half the area under the curve filled. Our approach would more pessimistically (and in this case correctly) produce zero area under the curve.

Table 3. Performance of all single-stage detector SUTs under four different performance metrics. In each cell, colorization from left to right is for performance on ROC_B, Roboticizer, Pathifier, then Closed Loop Control. Color bars are as in Table 5.



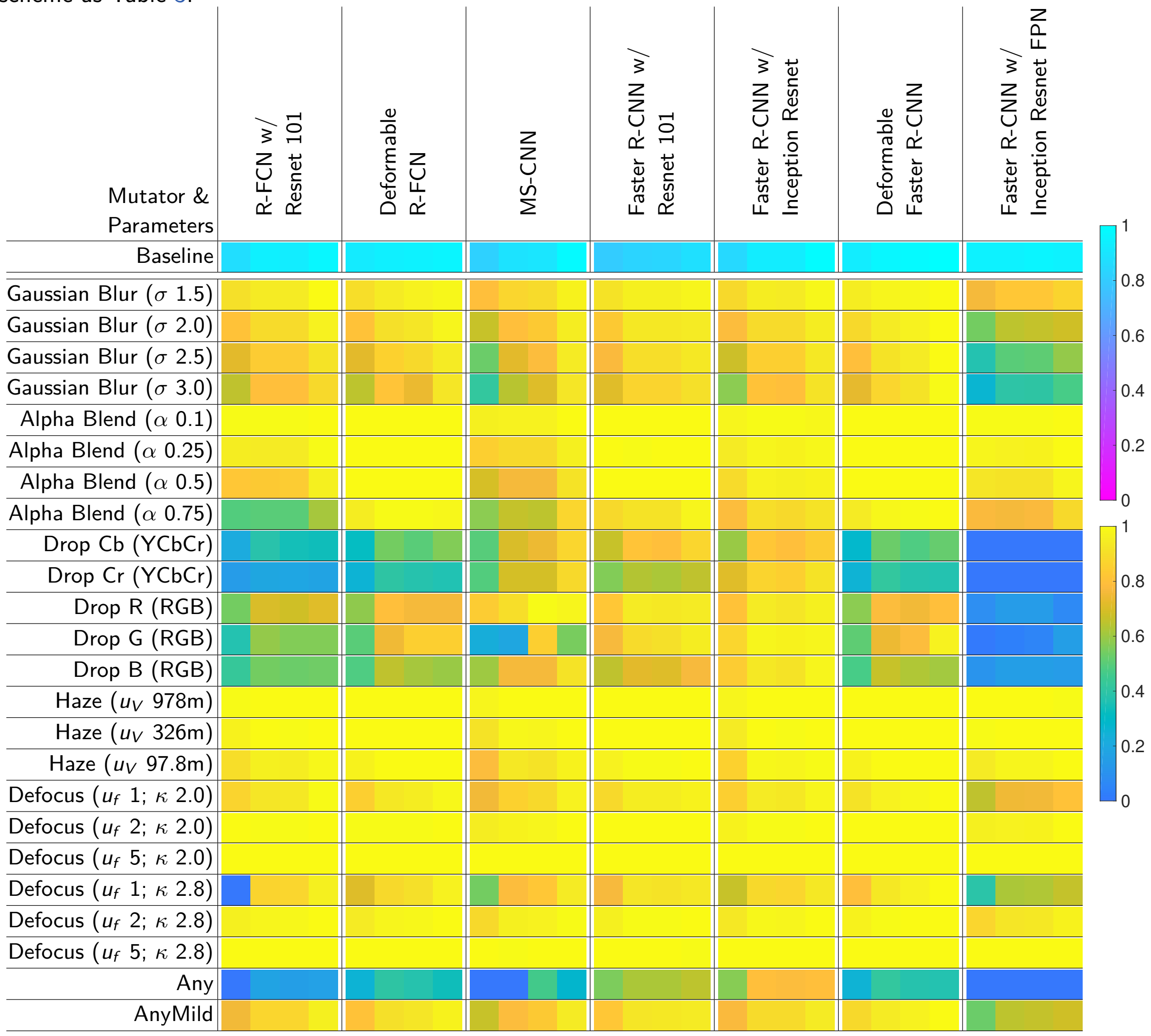
again, performances go up slightly, but rankings and robustness trends are maintained. Since the localization process for the 2D ground truth is the same as that used for detections from the Systems Under Test, this indicates that inaccuracies in this process are not a major factor in performance.

Tables 3 and 4 also aggregate a metric from each context level for easier comparison of overall trends. The two-stage detectors in our set generally perform better than the single-stage detectors on unmodified images, with the notable exception of RetinaNet, which scores best overall. The ranking of baseline scores is fairly stable across context levels, though performance values grow closer together.

The two-stage detectors also show generally greater robustness trends. All of the single-stage detectors' performance collapses under the channel dropout, though YOLOv3 and RetinaNet perform better than the rest. The Faster R-CNN architectures show the greatest robustness both to mild and severe mutations, with Faster R-CNN with Inception ResNet v2 showing much less drop in performance than other systems in almost all cases. The FPN version of Faster R-CNN with Inception ResNet v2is strikingly less robust than the standard version, though the difference could easily be more due to differences in the implementations' default training configuration than to the different architectures.

Within Tables 3 and 4, performance generally increases when going from left to right within each cell (colors growing lighter for baseline scores and warmer for robustness scores), as the additional

Table 4. Performance of all two-stage detector SUTs under four different performance metrics, using the same scheme as Table 3.



context filters out cases that are less relevant to system behavior and also tend to be more difficult. Performances thereby become more compressed at higher context levels, mitigating some of the more severe drops in performance under mutation.

Most notably, the choice of best-performing classifier may change as more context is added, depending on one's level of concern with robustness, and to what level of severity. For instance, at the outset, the FPN-based detectors appear the strongest performers, with the highest baseline scores. RetinaNet and YOLOv3's scores (particularly at the lowest context level), suggest a single-stage detector with a lower computational budget might be sufficient. Once robustness is considered, the choice between these and Faster R-CNN with Inception ResNet v2 becomes less clear-cut; YOLOv3 and RetinaNet still do slightly better on mild mutations, but do much worse on severe mutations. Then as more context is added, baseline scores draw much closer together, and Faster R-CNN with Inception ResNet v2 passes those two on mild mutations as well. However, if mild mutations are your biggest concern, Deformable Faster R-CNN performs best on them at all contexts levels, though it does not lead at any level on severe mutations.

Table 5. Performance under ROC_A, based on area under a traditional ROC curve. The color bar at top right is for baseline scores, and the one at bottom right is for robustness with respect to each baseline score.

Mutator & Parameters	SSD w/ MobileNet	SSD w/ Inception	YOLOv2	YOLOv3	RetinaNet FPN	R-FCN w/ Resnet 101	Deformable R-FCN	MS-CNN	Faster R-CNN w/ Resnet 101	Faster R-CNN w/ Inception Resnet	Deformable Faster R-CNN	Faster R-CNN w/ Inception Resnet FPN
Baseline	**0.39**	**0.35**	**0.52**	**0.78**	**0.88**	**0.74**	**0.81**	**0.69**	**0.70**	**0.74**	**0.82**	**0.83**
Gaussian Blur (σ 1.5)	0.25	0.29	0.40	0.70	0.77	0.66	0.72	0.53	0.64	0.65	0.77	0.62
Gaussian Blur (σ 2.0)	0.16	0.24	0.30	0.61	0.66	0.58	0.64	0.44	0.58	0.57	0.70	0.44
Gaussian Blur (σ 2.5)	0.10	0.23	0.24	0.52	0.52	0.51	0.57	0.34	0.54	0.50	0.64	0.30
Gaussian Blur (σ 3.0)	0.07	0.17	0.18	0.44	0.39	0.46	0.51	0.27	0.50	0.42	0.58	0.21
Alpha Blend (α 0.1)	0.39	0.33	0.48	0.78	0.87	0.73	0.81	0.65	0.70	0.73	0.82	0.83
Alpha Blend (α 0.25)	0.34	0.29	0.46	0.77	0.86	0.71	0.81	0.55	0.70	0.70	0.82	0.81
Alpha Blend (α 0.5)	0.10	0.15	0.38	0.73	0.82	0.62	0.80	0.41	0.70	0.66	0.81	0.77
Alpha Blend (α 0.75)	0.00	0.00	0.17	0.65	0.73	0.35	0.75	0.35	0.61	0.57	0.76	0.64
Drop Cb (YCbCr)	0.01	0.00	0.00	0.29	0.01	0.13	0.24	0.31	0.46	0.43	0.20	0.00
Drop Cr (YCbCr)	0.00	0.00	0.07	0.26	0.19	0.09	0.20	0.33	0.39	0.52	0.18	0.00
Drop R (RGB)	0.10	0.01	0.00	0.39	0.22	0.38	0.46	0.51	0.57	0.59	0.44	0.07
Drop G (RGB)	0.04	0.00	0.06	0.24	0.54	0.26	0.39	0.11	0.53	0.64	0.39	0.01
Drop B (RGB)	0.07	0.07	0.09	0.41	0.46	0.30	0.38	0.42	0.45	0.61	0.37	0.09
Haze (u_V 978m)	0.39	0.34	0.51	0.78	0.87	0.74	0.81	0.67	0.70	0.72	0.82	0.83
Haze (u_V 326m)	0.38	0.32	0.51	0.78	0.87	0.73	0.81	0.62	0.70	0.69	0.82	0.82
Haze (u_V 97.8m)	0.26	0.22	0.45	0.74	0.83	0.68	0.78	0.50	0.68	0.64	0.79	0.78
Defocus (u_f 1; κ 2.0)	0.22	0.23	0.37	0.67	0.73	0.62	0.68	0.50	0.61	0.62	0.74	0.53
Defocus (u_f 2; κ 2.0)	0.37	0.33	0.50	0.77	0.86	0.73	0.80	0.64	0.70	0.72	0.82	0.79
Defocus (u_f 5; κ 2.0)	0.39	0.35	0.52	0.78	0.87	0.74	0.81	0.69	0.70	0.74	0.82	0.83
Defocus (u_f 1; κ 2.8)	0.14	0.19	0.00	0.00	0.51	0.00	0.56	0.36	0.53	0.49	0.63	0.31
Defocus (u_f 2; κ 2.8)	0.32	0.30	0.46	0.74	0.82	0.70	0.76	0.59	0.67	0.69	0.79	0.72
Defocus (u_f 5; κ 2.8)	0.39	0.35	0.52	0.78	0.87	0.74	0.81	0.69	0.70	0.74	0.82	0.83
Any	0.00	0.00	0.00	0.00	0.00	0.00	0.19	0.00	0.39	0.41	0.17	0.00
AnyMild	0.16	0.20	0.28	0.59	0.65	0.56	0.63	0.44	0.58	0.57	0.70	0.43

Table 6. ROC_B performance metric.

Mutator & Parameters	SSD w/ MobileNet	SSD w/ Inception	YOLOv2	YOLOv3	RetinaNet FPN	R-FCN w/ Resnet 101	Deformable R-FCN	MS-CNN	Faster R-CNN w/ Resnet 101	Faster R-CNN w/ Inception Resnet	Deformable Faster R-CNN	Faster R-CNN w/ Inception Resnet FPN
Baseline	**0.43**	**0.37**	**0.57**	**0.80**	**0.89**	**0.77**	**0.81**	**0.73**	**0.71**	**0.75**	**0.83**	**0.84**
Gaussian Blur (σ 1.5)	0.27	0.30	0.46	0.72	0.79	0.70	0.73	0.58	0.65	0.66	0.78	0.64
Gaussian Blur (σ 2.0)	0.17	0.26	0.35	0.63	0.69	0.62	0.65	0.48	0.59	0.58	0.72	0.45
Gaussian Blur (σ 2.5)	0.11	0.26	0.28	0.55	0.55	0.55	0.58	0.38	0.55	0.50	0.66	0.31
Gaussian Blur (σ 3.0)	0.08	0.19	0.22	0.47	0.42	0.50	0.53	0.30	0.51	0.43	0.60	0.22
Alpha Blend (α 0.1)	0.43	0.35	0.53	0.80	0.88	0.76	0.81	0.69	0.71	0.73	0.83	0.83
Alpha Blend (α 0.25)	0.38	0.31	0.50	0.78	0.87	0.73	0.81	0.61	0.71	0.70	0.83	0.82
Alpha Blend (α 0.5)	0.12	0.16	0.43	0.75	0.83	0.64	0.81	0.50	0.71	0.66	0.82	0.78
Alpha Blend (α 0.75)	0.00	0.00	0.21	0.67	0.75	0.37	0.77	0.41	0.62	0.58	0.78	0.64
Drop Cb (YCbCr)	0.02	0.00	0.00	0.32	0.01	0.16	0.25	0.36	0.47	0.44	0.23	0.00
Drop Cr (YCbCr)	0.00	0.00	0.09	0.27	0.20	0.10	0.21	0.35	0.40	0.53	0.20	0.00
Drop R (RGB)	0.11	0.02	0.00	0.42	0.23	0.42	0.47	0.61	0.58	0.60	0.47	0.07
Drop G (RGB)	0.03	0.01	0.08	0.26	0.56	0.28	0.40	0.16	0.54	0.65	0.42	0.01
Drop B (RGB)	0.07	0.07	0.13	0.43	0.47	0.33	0.39	0.44	0.46	0.62	0.39	0.09
Haze (u_V 978m)	0.43	0.36	0.56	0.80	0.89	0.76	0.81	0.71	0.71	0.73	0.83	0.83
Haze (u_V 326m)	0.42	0.34	0.55	0.79	0.88	0.74	0.81	0.67	0.71	0.70	0.82	0.83
Haze (u_V 97.8m)	0.29	0.23	0.49	0.76	0.84	0.69	0.79	0.56	0.68	0.64	0.80	0.79
Defocus (u_f 1; κ 2.0)	0.24	0.23	0.43	0.69	0.75	0.66	0.69	0.55	0.62	0.63	0.75	0.54
Defocus (u_f 2; κ 2.0)	0.41	0.33	0.56	0.79	0.87	0.76	0.80	0.69	0.70	0.73	0.82	0.80
Defocus (u_f 5; κ 2.0)	0.43	0.37	0.57	0.80	0.89	0.77	0.81	0.73	0.71	0.75	0.83	0.84
Defocus (u_f 1; κ 2.8)	0.15	0.19	0.00	0.00	0.54	0.00	0.57	0.39	0.54	0.50	0.65	0.32
Defocus (u_f 2; κ 2.8)	0.37	0.30	0.52	0.76	0.83	0.73	0.76	0.64	0.67	0.70	0.80	0.73
Defocus (u_f 5; κ 2.8)	0.43	0.37	0.57	0.80	0.89	0.77	0.81	0.72	0.70	0.75	0.83	0.84
Any	0.00	0.00	0.00	0.00	0.00	0.00	0.21	0.00	0.39	0.42	0.20	0.00
AnyMild	0.17	0.20	0.32	0.62	0.66	0.58	0.65	0.48	0.59	0.58	0.72	0.44

Table 7. Roboticizer performance metric with 3D ground truth.

Mutator & Parameters	SSD w/ MobileNet	SSD w/ Inception	YOLOv2	YOLOv3	RetinaNet FPN	R-FCN w/ Resnet 101	Deformable R-FCN	MS-CNN	Faster R-CNN w/ Resnet 101	Faster R-CNN w/ Inception Resnet	Deformable Faster R-CNN	Faster R-CNN w/ Inception Resnet FPN
Baseline	**0.55**	**0.47**	**0.79**	**0.83**	**0.88**	**0.82**	**0.82**	**0.77**	**0.73**	**0.81**	**0.85**	**0.83**
Gaussian Blur (σ 1.5)	0.44	0.42	0.70	0.79	0.82	0.77	0.78	0.67	0.70	0.77	0.83	0.67
Gaussian Blur (σ 2.0)	0.32	0.40	0.66	0.72	0.74	0.73	0.74	0.61	0.67	0.72	0.80	0.54
Gaussian Blur (σ 2.5)	0.20	0.41	0.63	0.66	0.66	0.69	0.71	0.55	0.65	0.69	0.77	0.41
Gaussian Blur (σ 3.0)	0.14	0.37	0.57	0.60	0.56	0.66	0.67	0.49	0.63	0.66	0.73	0.33
Alpha Blend (α 0.1)	0.55	0.42	0.62	0.82	0.87	0.81	0.82	0.75	0.72	0.81	0.85	0.82
Alpha Blend (α 0.25)	0.48	0.37	0.60	0.81	0.87	0.77	0.82	0.68	0.72	0.79	0.85	0.80
Alpha Blend (α 0.5)	0.17	0.27	0.51	0.79	0.85	0.68	0.82	0.59	0.72	0.79	0.84	0.75
Alpha Blend (α 0.75)	0.00	0.00	0.33	0.70	0.77	0.41	0.81	0.51	0.66	0.73	0.83	0.63
Drop Cb (YCbCr)	0.07	0.00	0.00	0.46	0.03	0.31	0.45	0.54	0.59	0.67	0.44	0.00
Drop Cr (YCbCr)	0.02	0.00	0.40	0.40	0.33	0.16	0.33	0.53	0.46	0.70	0.35	0.00
Drop R (RGB)	0.25	0.07	0.01	0.56	0.34	0.58	0.65	0.69	0.68	0.76	0.66	0.12
Drop G (RGB)	0.06	0.04	0.37	0.40	0.67	0.48	0.61	0.14	0.65	0.79	0.63	0.02
Drop B (RGB)	0.14	0.16	0.48	0.55	0.62	0.44	0.53	0.59	0.52	0.76	0.56	0.12
Haze (u_V 978m)	0.55	0.47	0.78	0.83	0.88	0.82	0.82	0.77	0.72	0.81	0.85	0.83
Haze (u_V 326m)	0.55	0.46	0.77	0.83	0.88	0.82	0.82	0.76	0.72	0.81	0.85	0.83
Haze (u_V 97.8m)	0.51	0.40	0.72	0.82	0.87	0.78	0.82	0.72	0.72	0.80	0.85	0.81
Defocus (u_f 1; κ 2.0)	0.42	0.40	0.71	0.78	0.80	0.76	0.77	0.66	0.69	0.76	0.82	0.62
Defocus (u_f 2; κ 2.0)	0.54	0.46	0.78	0.82	0.86	0.81	0.82	0.75	0.72	0.81	0.84	0.80
Defocus (u_f 5; κ 2.0)	0.55	0.47	0.79	0.83	0.88	0.82	0.82	0.77	0.73	0.81	0.84	0.83
Defocus (u_f 1; κ 2.8)	0.29	0.38	0.03	0.02	0.72	0.71	0.73	0.60	0.67	0.71	0.79	0.51
Defocus (u_f 2; κ 2.8)	0.53	0.44	0.77	0.82	0.85	0.80	0.80	0.74	0.72	0.80	0.84	0.77
Defocus (u_f 5; κ 2.8)	0.54	0.47	0.79	0.83	0.88	0.82	0.82	0.77	0.72	0.81	0.84	0.83
Any	0.00	0.00	0.00	0.02	0.03	0.14	0.33	0.00	0.45	0.65	0.35	0.00
AnyMild	0.31	0.30	0.46	0.72	0.74	0.71	0.74	0.61	0.67	0.72	0.80	0.53

Table 8. Roboticizer with 2D ground truth.

Mutator & Parameters	SSD w/ MobileNet	SSD w/ Inception	YOLOv2	YOLOv3	RetinaNet FPN	R-FCN w/ Resnet 101	Deformable R-FCN	MS-CNN	Faster R-CNN w/ Resnet 101	Faster R-CNN w/ Inception Resnet	Deformable Faster R-CNN	Faster R-CNN w/ Inception Resnet FPN
Baseline	**0.58**	**0.50**	**0.83**	**0.88**	**0.94**	**0.86**	**0.87**	**0.82**	**0.76**	**0.86**	**0.90**	**0.88**
Gaussian Blur (σ 1.5)	0.47	0.44	0.76	0.83	0.87	0.81	0.83	0.73	0.73	0.81	0.88	0.72
Gaussian Blur (σ 2.0)	0.35	0.42	0.72	0.76	0.80	0.76	0.79	0.67	0.71	0.76	0.85	0.57
Gaussian Blur (σ 2.5)	0.22	0.43	0.68	0.70	0.70	0.72	0.76	0.62	0.68	0.72	0.82	0.43
Gaussian Blur (σ 3.0)	0.16	0.39	0.61	0.63	0.60	0.68	0.72	0.56	0.66	0.68	0.78	0.34
Alpha Blend (α 0.1)	0.57	0.45	0.66	0.87	0.93	0.85	0.87	0.79	0.76	0.85	0.90	0.87
Alpha Blend (α 0.25)	0.49	0.39	0.63	0.86	0.93	0.81	0.87	0.72	0.76	0.84	0.90	0.85
Alpha Blend (α 0.5)	0.16	0.28	0.54	0.82	0.90	0.71	0.87	0.61	0.76	0.82	0.90	0.80
Alpha Blend (α 0.75)	0.00	0.00	0.34	0.73	0.81	0.42	0.86	0.52	0.69	0.76	0.87	0.66
Drop Cb (YCbCr)	0.07	0.00	0.00	0.47	0.04	0.31	0.45	0.55	0.60	0.70	0.45	0.00
Drop Cr (YCbCr)	0.02	0.00	0.40	0.40	0.32	0.16	0.33	0.52	0.46	0.73	0.36	0.00
Drop R (RGB)	0.28	0.08	0.01	0.56	0.36	0.58	0.67	0.73	0.71	0.80	0.68	0.12
Drop G (RGB)	0.06	0.05	0.37	0.40	0.71	0.49	0.65	0.15	0.69	0.84	0.65	0.02
Drop B (RGB)	0.14	0.16	0.48	0.56	0.63	0.44	0.54	0.59	0.53	0.79	0.57	0.12
Haze (u_V 978m)	0.58	0.50	0.82	0.87	0.94	0.86	0.87	0.81	0.76	0.85	0.90	0.88
Haze (u_V 326m)	0.58	0.48	0.81	0.87	0.94	0.86	0.87	0.80	0.76	0.85	0.90	0.88
Haze (u_V 97.8m)	0.53	0.42	0.76	0.86	0.93	0.82	0.87	0.75	0.75	0.84	0.90	0.86
Defocus (u_f 1; κ 2.0)	0.46	0.43	0.77	0.82	0.86	0.80	0.82	0.73	0.72	0.80	0.86	0.66
Defocus (u_f 2; κ 2.0)	0.56	0.48	0.82	0.87	0.92	0.86	0.87	0.80	0.76	0.85	0.90	0.85
Defocus (u_f 5; κ 2.0)	0.58	0.50	0.83	0.87	0.94	0.86	0.87	0.81	0.76	0.85	0.90	0.88
Defocus (u_f 1; κ 2.8)	0.32	0.40	0.03	0.03	0.77	0.74	0.78	0.67	0.70	0.75	0.84	0.54
Defocus (u_f 2; κ 2.8)	0.55	0.47	0.82	0.87	0.90	0.85	0.85	0.79	0.76	0.84	0.89	0.82
Defocus (u_f 5; κ 2.8)	0.57	0.49	0.83	0.87	0.94	0.86	0.87	0.81	0.76	0.85	0.90	0.88
Any	0.00	0.00	0.00	0.03	0.03	0.14	0.33	0.00	0.46	0.67	0.36	0.00
AnyMild	0.31	0.31	0.51	0.76	0.80	0.75	0.79	0.67	0.70	0.76	0.85	0.57

Table 9. Pathifier performance metric.

Mutator & Parameters	SSD w/ MobileNet	SSD w/ Inception	YOLOv2	YOLOv3	RetinaNet FPN	R-FCN w/ Resnet 101	Deformable R-FCN	MS-CNN	Faster R-CNN w/ Resnet 101	Faster R-CNN w/ Inception Resnet	Deformable Faster R-CNN	Faster R-CNN w/ Inception Resnet FPN
Baseline	**0.62**	**0.56**	**0.88**	**0.87**	**0.95**	**0.88**	**0.89**	**0.85**	**0.79**	**0.88**	**0.93**	**0.90**
Gaussian Blur (σ 1.5)	0.47	0.44	0.83	0.85	0.89	0.83	0.86	0.76	0.76	0.83	0.91	0.74
Gaussian Blur (σ 2.0)	0.36	0.43	0.79	0.79	0.82	0.79	0.83	0.70	0.74	0.78	0.90	0.60
Gaussian Blur (σ 2.5)	0.26	0.44	0.77	0.72	0.73	0.74	0.79	0.66	0.71	0.74	0.87	0.46
Gaussian Blur (σ 3.0)	0.36	0.43	0.72	0.66	0.65	0.70	0.65	0.60	0.68	0.69	0.84	0.36
Alpha Blend (α 0.1)	0.61	0.51	0.72	0.86	0.95	0.87	0.89	0.82	0.79	0.87	0.93	0.89
Alpha Blend (α 0.25)	0.55	0.44	0.71	0.83	0.94	0.83	0.89	0.75	0.79	0.85	0.93	0.87
Alpha Blend (α 0.5)	0.31	0.30	0.62	0.80	0.91	0.74	0.89	0.64	0.79	0.84	0.92	0.83
Alpha Blend (α 0.75)	0.03	0.00	0.43	0.73	0.83	0.44	0.89	0.55	0.72	0.78	0.92	0.69
Drop Cb (YCbCr)	0.20	0.00	0.00	0.49	0.03	0.30	0.45	0.63	0.63	0.70	0.44	0.00
Drop Cr (YCbCr)	0.17	0.00	0.52	0.41	0.33	0.16	0.33	0.58	0.49	0.74	0.35	0.00
Drop R (RGB)	0.48	0.13	0.05	0.58	0.39	0.60	0.68	0.84	0.74	0.81	0.70	0.12
Drop G (RGB)	0.15	0.11	0.47	0.43	0.75	0.49	0.75	0.72	0.72	0.85	0.72	0.04
Drop B (RGB)	0.41	0.20	0.60	0.58	0.65	0.46	0.55	0.64	0.56	0.80	0.58	0.14
Haze (u_V 978m)	0.62	0.55	0.87	0.87	0.95	0.88	0.89	0.84	0.79	0.88	0.93	0.90
Haze (u_V 326m)	0.62	0.54	0.86	0.85	0.95	0.87	0.89	0.83	0.79	0.87	0.93	0.90
Haze (u_V 97.8m)	0.54	0.46	0.82	0.82	0.94	0.84	0.89	0.78	0.79	0.86	0.93	0.88
Defocus (u_f 1; κ 2.0)	0.46	0.44	0.83	0.84	0.87	0.82	0.85	0.75	0.75	0.82	0.91	0.68
Defocus (u_f 2; κ 2.0)	0.59	0.51	0.87	0.87	0.94	0.87	0.89	0.83	0.79	0.87	0.93	0.88
Defocus (u_f 5; κ 2.0)	0.61	0.55	0.88	0.87	0.95	0.88	0.89	0.85	0.79	0.88	0.93	0.90
Defocus (u_f 1; κ 2.8)	0.56	0.56	0.12	0.03	0.79	0.76	0.81	0.69	0.73	0.77	0.89	0.56
Defocus (u_f 2; κ 2.8)	0.62	0.56	0.87	0.87	0.92	0.87	0.88	0.82	0.78	0.86	0.92	0.84
Defocus (u_f 5; κ 2.8)	0.62	0.56	0.87	0.87	0.95	0.88	0.89	0.84	0.79	0.88	0.93	0.90
Any	0.00	0.00	0.00	0.03	0.03	0.14	0.33	0.39	0.49	0.68	0.35	0.00
AnyMild	0.34	0.32	0.56	0.78	0.81	0.76	0.83	0.70	0.74	0.78	0.89	0.59

Table 10. Closed Loop Control performance metric.

Mutator & Parameters	SSD w/ MobileNet	SSD w/ Inception	YOLOv2	YOLOv3	RetinaNet FPN	R-FCN w/ Resnet 101	Deformable R-FCN	MS-CNN	Faster R-CNN w/ Resnet 101	Faster R-CNN w/ Inception Resnet	Deformable Faster R-CNN	Faster R-CNN w/ Inception Resnet FPN
Baseline	**0.73**	**0.67**	**0.86**	**0.85**	**0.89**	**0.87**	**0.85**	**0.87**	**0.79**	**0.88**	**0.89**	**0.84**
Gaussian Blur (σ 1.5)	0.66	0.56	0.84	0.80	0.86	0.86	0.84	0.84	0.77	0.87	0.89	0.73
Gaussian Blur (σ 2.0)	0.59	0.53	0.81	0.75	0.83	0.83	0.83	0.83	0.74	0.84	0.89	0.57
Gaussian Blur (σ 2.5)	0.52	0.57	0.78	0.71	0.79	0.79	0.80	0.82	0.73	0.82	0.89	0.49
Gaussian Blur (σ 3.0)	0.52	0.53	0.75	0.66	0.76	0.77	0.79	0.80	0.71	0.80	0.89	0.39
Alpha Blend (α 0.1)	0.72	0.58	0.81	0.85	0.89	0.86	0.85	0.86	0.78	0.87	0.89	0.84
Alpha Blend (α 0.25)	0.67	0.48	0.81	0.84	0.89	0.85	0.85	0.83	0.78	0.87	0.89	0.84
Alpha Blend (α 0.5)	0.61	0.29	0.80	0.85	0.89	0.83	0.85	0.79	0.78	0.86	0.89	0.82
Alpha Blend (α 0.75)	0.44	0.00	0.62	0.77	0.87	0.53	0.84	0.75	0.77	0.81	0.89	0.74
Drop Cb (YCbCr)	0.49	0.01	0.00	0.53	0.05	0.29	0.47	0.76	0.68	0.74	0.47	0.00
Drop Cr (YCbCr)	0.36	0.01	0.59	0.41	0.50	0.15	0.31	0.77	0.51	0.80	0.33	0.00
Drop R (RGB)	0.64	0.22	0.05	0.60	0.56	0.62	0.64	0.85	0.74	0.84	0.72	0.06
Drop G (RGB)	0.32	0.25	0.75	0.41	0.81	0.48	0.72	0.47	0.74	0.86	0.86	0.13
Drop B (RGB)	0.51	0.23	0.66	0.59	0.77	0.46	0.51	0.79	0.60	0.86	0.54	0.12
Haze (u_V 978m)	0.72	0.64	0.86	0.85	0.89	0.86	0.85	0.87	0.78	0.88	0.89	0.83
Haze (u_V 326m)	0.72	0.59	0.85	0.85	0.89	0.86	0.85	0.87	0.78	0.88	0.89	0.84
Haze (u_V 97.8m)	0.68	0.48	0.85	0.85	0.89	0.85	0.85	0.84	0.78	0.87	0.89	0.84
Defocus (u_f 1; κ 2.0)	0.65	0.55	0.82	0.77	0.85	0.86	0.84	0.84	0.76	0.86	0.89	0.68
Defocus (u_f 2; κ 2.0)	0.72	0.63	0.86	0.84	0.89	0.86	0.85	0.87	0.78	0.88	0.89	0.84
Defocus (u_f 5; κ 2.0)	0.73	0.66	0.85	0.85	0.89	0.87	0.85	0.87	0.78	0.88	0.89	0.84
Defocus (u_f 1; κ 2.8)	0.57	0.53	0.07	0.02	0.81	0.83	0.82	0.82	0.73	0.82	0.89	0.55
Defocus (u_f 2; κ 2.8)	0.72	0.61	0.86	0.84	0.89	0.86	0.85	0.86	0.78	0.88	0.89	0.82
Defocus (u_f 5; κ 2.8)	0.73	0.66	0.85	0.85	0.89	0.87	0.85	0.86	0.78	0.88	0.89	0.84
Any	0.15	0.00	0.00	0.02	0.03	0.14	0.29	0.24	0.51	0.70	0.33	0.00
AnyMild	0.51	0.40	0.72	0.75	0.83	0.83	0.83	0.82	0.73	0.83	0.89	0.57

10. Discussion

By taking a step back, we can look at the larger implications as well as any threats to validity. Component-level evaluation facilitates the comparison of many models and has enabled fast progress through instant feedback and sometimes the prestige of a public ranking. However, our work shows that these improvements may not always propagate to the end system. These models are only parts of a larger system, and simulating system-level context can avoid suboptimal decisions, despite limitations on the approximation.

While we advocate adding levels of context earlier in the trade-off and design process, *it can take significant resources to define the other aspects of the system.* If these components are not accurately modeled, the translation from standard, independent, image-level detection metrics, such as mean Average Precision will not translate accurately into higher level business case and safety metrics. This can mislead design decisions and trade-offs. However, we argue that bringing in higher level system metrics is always useful. Over the design life-cycle, modeling accuracy can be refined, and this iterative refinement forces integration to the front of the developer's mind, which is important for effective engineering. This will facilitate the processes of integration and performance verification by minimizing later surprises.

Our approach is limited to data collections that are safe and practical to collect. This condition goes hand in hand with the evidence that, even with large datasets, it is difficult to capture all scenes and relevant challenges, including myriad potential failure cases. Robot sensor data are also highly temporally correlated and can be expensive to collect and annotate, both of which make it challenging to achieve broad data variety. There is evidence that these properties lend themselves to non-robust features, which weakness exposes models to adversarial attacks. However, we believe that collecting real-world data of these edge case and distribution-shift scenarios is important, independent of the final perception evaluation method. Thinking through the safety of these scenarios out in the field during data collection helps guide system decisions and robot control behavior.

Adversarial attacks are not captured by our method currently. As described in Section 2.3, methods specifically optimizing image-mutations to cause failures in classification networks can often succeed with imperceptible changes. However, such attacks have not been as successful on networks for detection or without access to system internals. Model-extraction attacks do not seem like a large risk to fielded, embedded robotics systems either. Nonetheless, our method can easily incorporate adversarial attacks. While such attacks seem unlikely, it is hard to quantify the likelihood of any of these potential failure modes. Why not simply look at the worst-case performance? It can at least provide a lower bound on performance, and if we achieve our goals with worst-case performance, engineers deliver a system that is better performing than predicted. However, focusing on just the worst-case performance can result in the selection of an algorithm with lower average efficiency.

If one has complete knowledge of failure modes and their likelihood, training on such mutations or attacks can facilitate handling them directly. This will expand the data manifold to cover all the potential dataset shifts and significantly improve robustness. Alternatively, they can be detected and mitigated through alerting operators or supervisors. We think augmentation and training on attacks can help, but there is a training-time cost and nominal performance cost for this. We simply provide another tool for massaging the different system trade-offs: It should be used in tandem with augmentation and failure mode detection. There is also a high chance that there still exists a mutation that is not known—leaving some known mutations held out for use in our method can offer some insight on how a system performs in these unknown conditions.

Another argument is *if one is going to simulate that level of context, one might as well move to full simulation.* This provides full control over scenarios, especially as robots gain more decision making power. Our method can only handle slowing down of the robot. It cannot model the robot choosing a different path, or how the other agents sharing the world may react differently. Annotating real-world data can also introduce errors, which may not be an issue with a well-engineered simulation. However, fully qualifying and verifying a robot in simulation can be dangerous, as there are many examples of methods that do not translate well from simulation to reality, as it is infeasible to completely model the real world. One should capture as many scenarios in the field as possible, which enables our methods to work well, and also use simulation in addition to real-world data. Trustworthy ground truth is imperative to developing robotic systems, and significant effort needs to be in place to validate it.

Our mutations are only approximations of a subset of difficult cases (Zendel et al., 2015). Aside from it being unrealistic to accurately model all possible scenarios, the idea of holding out a subset of mutations from training augmentation and measuring robustness can be viewed as another type of hold-out set to test generalization and robustness at higher context levels. If two detectors have similar AUC metrics under impaired scenarios, we believe that our robustness score will better capture robustness to unknown unknowns.

In the same vein, *our chosen object detectors are only a subset of what is available and will be developed*: It is impractical to train and test all conceivable object detectors. We have chosen a number of popular detectors across complexity and age in hopes to show that our analysis methods are valuable across detectors. We hope others will start to include measures of robustness and system-level effectiveness when they develop new detectors.

Our benchmark problem is only one task on one dataset. We are using the largest available off-road, person-detection dataset that includes the required secondary information for contextual mutations. We hope that as more robotic applications are deployed in the real world, critical robustness analysis will be done at every stage for all tasks: from scene understanding in urban environments to odometry in underground applications.

11. Conclusions and future work

We introduced a new approach to robustness analysis, building on work to mutate images according to simulations of physical phenomena. We showed how to compare performance on safety-vs-efficiency trade-off curves, before and after mutation, to measure robustness across a range of algorithm parameters using a novel metric, Robustness ROC. We demonstrated the approach on a large dataset, uncovering patterns of algorithm robustness for differing architectures. We believe these techniques can allow developers to discover and address robustness issues much earlier in the design process. They can also enable deeper analysis of operational conditions, even without having a physical robot, whether due to lack of access or because the system has not yet been fielded or even built.

We found that incorporating application context in analysis can both change rankings and connect results more easily to intuitive system-level behavior statistics. Notably, our work showed how decisions regarding the best algorithm to apply would change based on robustness priorities—while recent improvements in fast, single-stage detectors have allowed them to often out-perform two-stage detectors in the nominal case, our Systems Under Test still show a performance gap in suboptimal conditions. This result shows the importance of our approach both during component selection and future performance verification.

There are several opportunities to extend this work for greater reach. One is a deep investigation of training methods, discussed in Section 2.1, to imbue resilience to data shifts. Additionally, there are many possibilities for new mutations to test on, including a variety of weather conditions and problems with camera imaging. One interesting challenge for time-varying phenomena is how to ensure temporal consistency of effects across video (e.g., ensuring everything moves appropriately from frame to frame, even as the robot moves); this seems an area where tools from full simulation could be useful. Designers of real systems have more parameters than just detector sensitivity. For these systems, our work could be combined with multi-objective optimization to find the trade-off curve and its area. If the designer has additional objectives besides safety and efficiency, then our method would require at least the following modifications.

- Algorithm 1 would need to consider nearest neighbors in each dimension
- Area under the curve would need to be replaced with volume or Lebesgue measure, depending on number of objectives

- Ratio of volumes may need to be reconsidered as number of dimensions grows.

Another possible direction would investigate how best to combine the results of many experiments. In several applications (e.g., outside of safety), average-case performance is more relevant than worst-case performance. This observation introduces challenges, however, in accurately estimating the likelihood of particular conditions. This analysis has also focused on a single mutation at a time, but some failures may only appear when mutations occur in combination, so effectively exploring that combinatorial space remains a challenge. Finally, all of this analysis pertains only directly to the mutations being applied, the "known unknowns". However, the robustness results presented here indicate rather consistent trends across mutations for a particular System Under Test. It should therefore be possible to extrapolate and reason about robustness to "unknown unknowns" as well.

Acknowledgments

This project was funded by the Test Resource Management Center (TRMC) and Test and Evaluation/ Science & Technology (T&E/S&T) Program through the U.S. Army Program Executive Office for Simulation, Training and Instrumentation (PEO STRI) under Contract No. W900KK-16-C-0006, "Robustness Inside-Out Testing (RIOT)." NAVAIR Public Release 2021-107. Distribution Statement A - "Approved for public release; distribution is unlimited". We also thank Nadine Trenton for help in preparing figures and other authors for making code public to enable such studies.

ORCID

Zachary Pezzementi https://orcid.org/0000-0002-1137-7378
Philip Koopman https://orcid.org/0000-0003-1658-2386

References

Altman, E. (1999). *Constrained Markov decision processes* (Vol. 7). CRC Press.

Amini, A., Gilitschenski, I., Phillips, J., Moseyko, J., Banerjee, R., Karaman, S., & Rus, D. (2020). Learning robust control policies for end-to-end autonomous driving from data-driven simulation. *IEEE Robotics and Automation Letters*, *5*(2), 1143–1150. https://doi.org/10.1109/lra.2020.2966414

Argo AI. (2019, June). *How we test.* Retrieved March 15, 2019, from https://www.argo.ai/how-we-test/

Ariizumi, R., Tesch, M., Kato, K., Choset, H., & Matsuno, F. (2016). Multiobjective optimization based on expensive robotic experiments under heteroscedastic noise. *IEEE Transactions on Robotics*, *33*(2), 468–483. https://doi.org/10.1109/tro.2016.2632739

Arroyo, S. D. P. (2013). *Modeling and applications of the focus cue in conventional digital cameras* (Doctoral dissertation). Universitat Rovira i Virgili. Tarragona, Spain. http://hdl.handle.net/10803/123829

Athalye, A., Engstrom, L., Ilyas, A., & Kwok, K. (2018). *Synthesizing robust adversarial examples.* arXiv: 1707.07397 [cs.CV].

Baidu AI. (2017). *Apollo simulation.* Retrieved November 14, 2017, from https://apollo.auto/

Barron, J. T., & Poole, B. (2016). The fast bilateral solver. In B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.), *Lecture Notes in Computer Science: Vol. 9907. Computer Vision – ECCV 2016* (pp. 617–632). Springer. https://doi.org/10.1007/978-3-319-46487-9_38

von Bernuth, A., Volk, G., & Bringmann, O. (2019). Simulating photo-realistic snow and fog on existing images for enhanced CNN training and evaluation. *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 41–46. https://doi.org/10.1109/itsc.2019.8917367

Bigelow, P. (2019, June). For self-driving progress, Aurora focuses inward. *Automotive News.* Retrieved March 15, 2019, from https://www.autonews.com/mobility-report/self-driving-progress-aurora-focuses-inward

Cai, Z., Fan, Q., Feris, R. S., & Vasconcelos, N. (2016). A unified multi-scale deep convolutional neural network for fast object detection. In B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.), *Lecture Notes in Computer Science: Vol. 9908. Computer Vision – ECCV 2016* (pp. 354–370). Springer. https://doi.org/10.1007/978-3-319-46493-0_22

Cubuk, E. D., Zoph, B., Mané, D., Vasudevan, V., & Le, Q. V. (2019). AutoAugment: Learning augmentation strategies from data. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 113–123. https://doi.org/10.1109/cvpr.2019.00020

Dai, J., Li, Y., He, K., & Sun, J. (2016). R-FCN: Object detection via region-based fully convolutional networks. *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 379–387.

Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., & Wei, Y. (2017). Deformable convolutional networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, 764–773. https://doi.org/10.1109/iccv.2017.89

Dalal, G., Dvijotham, K., Vecerik, M., Hester, T., Paduraru, C., & Tassa, Y. (2018). *Safe exploration in continuous action spaces*. arXiv: 1801.08757 [cs.AI].

Dima, C., Wellington, C., Moorehead, S., Lister, L., Campoy, J., Vallespi, C., Jung, B., Kise, M., & Bonefas, Z. (2011). PVS: A system for large scale outdoor perception performance evaluation. *2011 IEEE International Conference on Robotics and Automation (ICRA)*, 834–841. https://doi.org/10.1109/icra.2011.5980419

Dong, Y., Zhang, P., Wang, J., Liu, S., Sun, J., Hao, J., Wang, X., Wang, L., Dong, J., & Dai, T. (2020). An empirical study on correlation between coverage and robustness for deep neural networks. *2020 25th International Conference on Engineering of Complex Computer Systems (ICECCS)*, 73–82. https://doi.org/10.1109/iceccs51672.2020.00016

Dulac-Arnold, G., Levine, N., Mankowitz, D. J., Li, J., Paduraru, C., Gowal, S., & Hester, T. (2021). Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis. *Machine Learning*, *110*(9), 2419–2468. https://doi.org/10.1007/s10994-021-05961-4

Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., & Song, D. (2018). Robust physical-world attacks on deep learning visual classification. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 1625–1634. https://doi.org/10.1109/cvpr.2018.00175

Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Song, D. X., Kohno, T., Rahmati, A., Prakash, A., & Tramèr, F. (2018). *Note on attacking object detectors with adversarial stickers*. arXiv: 1712.08062 [cs.CR].

Gaidon, A., Wang, Q., Cabon, Y., & Vig, E. (2016). VirtualWorlds as proxy for multi-object tracking analysis. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4340–4349. https://doi.org/10.1109/cvpr.2016.470

Galloway, A., Golubeva, A., Tanay, T., Moussa, M., & Taylor, G. W. (2019). *Batch normalization is a cause of adversarial vulnerability*. arXiv: 1905.02161 [cs.LG].

Geirhos, R., Temme, C. R. M., Rauber, J., Schütt, H. H., Bethge, M., & Wichmann, F. A. (2018). Generalisation in humans and deep neural networks. *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 7549–7561.

Girshick, R., Radosavovic, I., Gkioxari, G., Dollár, P., & He, K. (2018). *Detectron* [Computer software]. GitHub. https://github.com/facebookresearch/detectron

Gu, K., Yang, B., Ngiam, J., Le, Q., & Shlens, J. (2019). *Using videos to evaluate image model robustness*. arXiv: 1904.10076 [cs.CV].

Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, 1321–1330.

He, K., Sun, J., & Tang, X. (2011). Single image haze removal using dark channel prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(12), 2341–2353. https://doi.org/10.1109/tpami.2010.168

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. https://doi.org/10.1109/cvpr.2016.90

Hendrycks, D., & Dietterich, T. (2019). *Benchmarking neural network robustness to common corruptions and perturbations*. arXiv: 1903.12261 [cs.LG].

Hendrycks, D., Mu, N., Cubuk, E. D., Zoph, B., Gilmer, J., & Lakshminarayanan, B. (2020). *AugMix: A simple data processing method to improve robustness and uncertainty*. arXiv: 1912.02781 [stat.ML].

Hollnagel, E. (2009). *The ETTO principle: Efficiency-thoroughness trade-off: Why things that go right sometimes go wrong*. Ashgate Publishing, Ltd. https://doi.org/10.1201/9781315616247

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). *MobileNets: Efficient convolutional neural networks for mobile vision applications*. arXiv: 1704.04861 [cs.CV].

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., & Murphy, K. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3296–3297. https://doi.org/10.1109/cvpr.2017.351

Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., & Madry, A. (2019). Adversarial examples are not bugs, they are features. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. B. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems* (pp. 125–136). http://dblp.uni-trier.de/db/conf/nips/nips2019.html#IlyasSTETM19

Johnson-Roberson, M., Barto, C., Mehta, R., Sridhar, S. N., Rosaen, K., & Vasudevan, R. (2017). Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 746–753. https://doi.org/10.1109/icra.2017.7989092

Kalra, N., & Paddock, S. M. (2016). Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, *94*, 182–193. https://doi.org/10.7249/rr1478

Kang, D., Sun, Y., Hendrycks, D., Brown, T., & Steinhardt, J. (2020). *Testing robustness against unforeseen adversaries.* arXiv: 1908.08016 [cs.LG].

Koopman, P., & Wagner, M. (2016). Challenges in autonomous vehicle testing and validation. *SAE International Journal of Transportation Safety*, *4*(1), 15–24. https://doi.org/10.4271/2016-01-0128

Kurakin, A., Goodfellow, I., & Bengio, S. (2018). Adversarial examples in the physical world. In R. V. Yampolskiy (Ed.), *Artificial intelligence safety and security.* https://doi.org/10.1201/9781351251389-8

Li, Z., Peng, C., Yu, G., Zhang, X., Deng, Y., & Sun, J. (2018). *DetNet: A backbone network for object detection.* arXiv: 1804.06215 [cs.CV].

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. *2017 IEEE International Conference on Computer Vision (ICCV)*, 2999–3007. https://doi.org/10.1109/ICCV.2017.324

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single shot MultiBox detector. In B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.), *Lecture Notes in Computer Science. Computer Vision – ECCV 2016* (pp. 21–37). Springer. https://doi.org/10.1007/978-3-319-46448-0_2

Madrigal, A. C. (2017, August). Inside Waymo's secret world for training self-driving cars. *The Atlantic.* https://www.theatlantic.com/amp/article/537648/

Meilland, M., Drummond, T., & Comport, A. I. (2013). A unified rolling shutter and motion blur model for 3D visual registration. *2013 IEEE International Conference on Computer Vision (ICCV)*, 2016–2023. https://doi.org/10.1109/iccv.2013.252

Menze, M., & Geiger, A. (2015). Object scene flow for autonomous vehicles. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3061–3070. https://doi.org/10.1109/cvpr.2015.7298925

Michaelis, C., Mitzkus, B., Geirhos, R., Rusak, E., Bringmann, O., Ecker, A. S., Bethge, M., & Brendel, W. (2020). *Benchmarking robustness in object detection: Autonomous driving when winter is coming.* arXiv: 1907.07484 [cs.CV].

Moorehead, S. J., Wellington, C. K., Gilmore, B. J., & Vallespi, C. (2012). Automating orchards: A system of autonomous tractors for orchard maintenance. *IEEE/RSJ International Conference on Intelligent Robots and Systems Workshop on Agricultural Robotics, Vilamoura, Portugal.*

Nvidia. (2018). *Nvidia Drive.* Retrieved March 20, 2018, from https://www.nvidia.com/en-us/self-driving-cars/drive-px/

Pei, K., Cao, Y., Yang, J., & Jana, S. (2017). DeepXplore: Automated whitebox testing of deep learning systems. *Proceedings of the 26th Symposium on Operating Systems Principles*, 1–18. https://doi.org/10.1145/3132747.3132785

Peng, X. B., Andrychowicz, M., Zaremba, W., & Abbeel, P. (2018). Sim-to-real transfer of robotic control with dynamics randomization. *2018 IEEE international conference on robotics and automation (ICRA)*, 3803–3810. https://doi.org/10.1109/icra.2018.8460528

Pezzementi, Z., Tabor, T., Hu, P., Chang, J. K., Ramanan, D., Wellington, C., Babu, B. P. W., & Herman, H. (2018). Comparing apples and oranges: Off-road pedestrian detection on the National Robotics Engineering Center agricultural person-detection dataset. *Journal of Field Robotics*, *35*(4), 545–563. https://doi.org/10.1002/rob.21760

Pezzementi, Z., Tabor, T., Yim, S., Chang, J. K., Drozd, B., Guttendorf, D., Wagner, M., & Koopman, P. (2018). Putting image manipulations in context: Robustness testing for safe perception. *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 1–8. https://doi.org/10.1109/ssrr.2018.8468619

Raghunathan, A., Xie, S. M., Yang, F., Duchi, J. C., & Liang, P. (2019). *Adversarial training can hurt generalization*. arXiv: 1906.06032 [cs.LG].

Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, faster, stronger. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6517–6525. https://doi.org/10.1109/cvpr.2017.690

Redmon, J., & Farhadi, A. (2018, April). *YOLOv3: An incremental improvement*. arXiv: 1804.02767 [cs.CV].

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6), 1137–1149. https://doi.org/10.1109/tpami.2016.2577031

Reynolds, H. (2019, June 5). Simulation: The invisible gatekeeper. *Medium*. https://medium.com/@UberATG/simulation-the-invisible-gatekeeper-e6ef84ea7647

Sakaridis, C., Dai, D., & Van Gool, L. (2018). Semantic foggy scene understanding with synthetic data. *International Journal of Computer Vision*, *126*(9), 973–992. https://doi.org/10.1007/s11263-018-1072-8

Stentz, A., Dima, C., Wellington, C., Herman, H., & Stager, D. (2002). A system for semi-autonomous tractor operations. *Autonomous Robots*, *13*(1), 87–104. https://doi.org/10.1023/A:1015634322857

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, Inception-ResNet and the impact of residual connections on learning. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 4278–4284.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2818–2826. https://doi.org/10.1109/cvpr.2016.308

Transportation, 49 C.F.R. § 393.52 - Brake performance (2019). Retrieved December 23, 2019, from https://www.law.cornell.edu/cfr/text/49/393.52

Veeravasarapu, V. S. R., Hota, R. N., Rothkopf, C., & Visvanathan, R. (2015). *Simulations for validation of vision systems*. arXiv: 1512.01030 [cs.CV].

Visibility. (2018). In *Wikipedia*. Retrieved February 7, 2018, from https://en.wikipedia.org/w/index.php?title=Visibility&oldid=819466260

Vogel, C., Schindler, K., & Roth, S. (2015). 3D scene flow estimation with a piecewise rigid scene model. *International Journal of Computer Vision*, *115*(1), 1–28. https://doi.org/10.1007/s11263-015-0806-0

Volk, G., Müller, S., von Bernuth, A., Hospach, D., & Bringmann, O. (2019). Towards robust CNN-based object detection through augmentation with synthetic rain variations. *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 285–292. https://doi.org/10.1109/itsc.2019.8917269

Wu, T., Tong, L., & Vorobeychik, Y. (2020). *Defending against physically realizable attacks on image classification*. arXiv: 1909.09552 [cs.LG].

Zendel, O., Murschitz, M., Humenberger, M., & Herzner, W. (2015). CV-HAZOP: Introducing test data validation for computer vision. *2015 IEEE International Conference on Computer Vision (ICCV)*, 2066–2074. https://doi.org/10.1109/iccv.2015.239

Zhang, H., & Wang, J. (2019). Towards adversarially robust object detection. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 421–430. https://doi.org/10.1109/iccv.2019.00051

Zhong, Z., Hu, Z., & Chen, X. (2020). Quantifying DNN model robustness to the real-world threats. *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 150–157. https://doi.org/10.1109/dsn48063.2020.00033

How to cite this article: Pezzementi, Z., Tabor, T., Chang, J. K., Tiernan, C., Drozd, B., Sample, E., Hazard, C., Wagner, M., & Koopman, P. (2021). Perception robustness testing at different levels of generality. *Field Robotics*, *1*, 253–286.

Appendix

Table A1. Glossary of terms.

Term	Definition
ROC	Receiver Operator Characteristic curve, a traditional trade-off curve for measuring the performance of object detectors
ROC_A	The standard formulation of the ROC curve that predominates in the previous literature
ROC_B	The standard ROC curve with two modifications: measure only false detections on images from negative logs, and evaluating the probability of a false detection in an image rather than the expected number of false detections per image.
S	A measure of safety, one axis of our generalized trade-off curves
E	A measure of efficiency, the other axis of our trade-off curves
RobROC	Robustness ROC, a derived trade-off curve from comparing behavior in normal conditions to that under some mutation. It is taken as a pessimistic lower bound on the performance across safety and efficiency objectives, but with the important guarantee that gains in one objective do not offset losses in another
SUT	System Under Test, an algorithm or entire system that is being tested for robustness