Submission version of paper accepted to IEEE SSRR 2018

Putting Image Manipulations in Context: Robustness Testing for Safe Perception

Zachary Pezzementi[1*], Trenton Tabor[1*], Samuel Yim[1], Jonathan K. Chang[1], Bill Drozd[1], David Guttendorf[1], Michael Wagner[1] and Philip Koopman[2]

***Abstract*— We introduce a method to evaluate the robustness of perception systems to the wide variety of conditions that a deployed system will encounter. Using person detection as a sample safety-critical application, we evaluate the robustness of several state-of-the-art perception systems to a variety of common image perturbations and degradations. We introduce two novel image perturbations that use "contextual information" (in the form of stereo image data) to perform more physically-realistic simulation of haze and defocus effects. For both standard and contextual mutations, we show cases where performance drops catastrophically in response to barely-perceptible changes. We also show how robustness to contextual mutators can be predicted without the associated contextual information in some cases.**

I. INTRODUCTION

With the rapid growth of autonomous systems being developed in safety-critical applications, we clearly need efficient ways to effectively test these systems and ensure they will be robust to the wide range of situations they will encounter. Ensuring safety through exhaustive real-world testing has been shown to be infeasible for most practical systems, requiring on the order of billions of test miles/hours of autonomous operation for a large-scale autonomous vehicle fleet [1], [2]. It seems much more promising to make use of the ever-growing availability of computational resources to expand on what is feasible with real-world testing alone.

One approach is to operate in simulation, like in the Virtual KITTI [3] and SYNTHIA [4] datasets and described by several autonomous driving companies. This allows tremendous control, but also requires crafting the entire world. Modeling some challenging conditions may require this level of environment knowledge, but we have found many relevant conditions that can be simulated from data collected by a typical robotic sensor package. Starting from real images reduces the burden of physical realism to the phenomena being modeled, which drastically reduces the scope and increases the applicability of the undertaking.

Another concern explored in recent papers is whether adversarial methods can be applied in the real world to produce objects that would be dangerous for robots. Kurakin et al. [5] showed that printing out adversarial perturbations of images still results in misclassification, providing examples of manipulated stop signs that are not recognized correctly. Athalye et al. [6] then showed that 3D printed objects could have the same effects from a wide range of views. We aim to first address the inputs that a robotic system is likely to encounter without adversarial intervention, so our focus is on naturally-occurring phenomena. Nonetheless, we show some examples of situations where nearly imperceptible image modifications can result in dramatic perception changes. Even in applications without malicious people trying to trick your system, the natural world may be adversarial enough.

Some previous work has investigated performance degradation of various algorithms (which we refer to as systems under test or SUTs) from simple image processing effects or "image mutations", such as Gaussian blur, additive noise, contrast enhancement, or JPEG compression [7], [8]. Karahan et al. [8] also examined the effect of occlusion by inserting black rectangles. Richard-Webster et al. [9] analyzed effects of some physical phenomena on rendered models, such as rotation, scaling, and linear occlusion. There has not been much focus in prior detection work on making these noise effects physically realistic though.

In this paper, we apply several simple image mutations to a large-scale dataset and evaluate the effects on performance for a safety-critical task, person detection. These include both procedural perturbations like blurring and randomized changes like additive noise, and others that remove image data. We then introduce "contextual mutators" that make use of environment geometry information to perform depth-based simulation of haze and defocus effects and compare these to the simple mutations. By evaluating a variety of different algorithms' performance, we show that the best performers 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.

We also show how performance on simple mutators can sometimes be used to predict performance on contextual mutators, allowing performance estimation even without context data. This also shows promise for extrapolation: robustness to a variety of well-understood conditions the system was not prepared for (known unknowns) provides some insight into robustness to other conditions that a tester has not thought of (unknown unknowns).

II. APPROACH

Our general approach to evaluate robustness is to take real images with associated ground truth for the desired SUT output, mutate them, and monitor how each SUT output

*Equal contribution

[1]National Robotics Engineering Center, Carnegie Mellon University, 10 40th St. Pittsburgh, PA, 15201, USA {pez, ttabor, syim, jchang, bdrozd, dguttendorf, mwagner}@nrec.ri.cmu.edu

[2]Electrical and Computer Engineering Department, Carnegie Mellon University koopman@cmu.edu

SUT	Base Library	Reference(s)
MS-CNN	Caffe	[11]
SSD w/ MobileNets	TensorFlow	[12]–[14]
SSD w/ Inception	TensorFlow	[12], [13], [15]
R-FCN w/ ResNet-101	TensorFlow	[12], [16], [17]
Faster R-CNN w/ ResNet-101	TensorFlow	[12], [17], [18]
Faster R-CNN w/ Inception ResNet v2	TensorFlow	[12], [18], [19]
Deformable R-FCN	MXNet	[16], [20]
Deformable Faster R-CNN	MXNet	[18], [20]

TABLE I: List of SUTs evaluated.

changes with the level of mutation. A robust SUT's output should ideally not change under mutation; practically, the change should be as small as possible. To ground the analysis in a domain relevant to robotics, we test on the NREC Agricultural Person Detection Dataset [10], a person detection benchmark for off-road mobile robots. This dataset includes nearly 10^5 labeled images for training and evaluation, derived from continuous video from stereo cameras.

We train all SUTs with default parameters from their respective publications or source code. The only factor that is evaluated on the validation set is training convergence, at which point we choose the training iteration with the best validation performance for evaluation. All results below are for the test set, as per the benchmark [10].

A. Performance Evaluation

We base our performance evaluation metrics the standard benchmark metrics, but we modify them to consider the following situation: A system has been developed and tuned using the training and validation set to choose a configuration for deployment. This includes the choice of a sensitivity for how strong a response needs to be to produce a detection. This choice was based on the conditions demonstrated in the training and validation data, but the system will be subjected to a wider variety of conditions in the following experiments.

We evaluate many different detectors with different confidence measures, so we standardize sensitivity based on expected false positive (FP) rates. Without confining our evaluation to a particular choice of sensitivity, we emulate the process of picking sensitivity thresholds based on the baseline ROC of the test set. For baseline performance, we sample locations on the ROC curve that correspond to FP rates ranging from 10^{-3} to 10^{-1}. In all ROCs for mutated data, we use the same sensitivities, rather than fixing them to the FP rates on mutated data. This reflects the fact that a system deployed with a chosen sensitivity can have its behavior change both in terms of true and false detections; moreover, this change can be much more dramatic than the ROC suggests. We use the same fixed sensitivities for computing average detection rate (ADR) on mutated images.

B. Systems Under Test (SUTs)

The full set of SUTs evaluated is shown in Table I.

Many modern convolutional neural networks for object detection use one of three meta-architectures: Single Shot Detector (SSD) [13], Faster R-CNN [18], and Region-based Fully Convolutional Network (R-FCN) [16]. SSD networks are fast; they directly predict classes and anchor offsets in a single feed-forward pass, making them more suitable for the computational constraints of embedded perception. Faster R-CNN uses a two-staged approach: a region proposal network generates proposed object regions, and a box classifier refines each of these proposals. R-FCN strikes a middle-ground in computation by moving the cropping to the last stage of the box classifier. This shares most of the box-classifier-specific feature computation across regions.

These meta-architectures can easily be paired with various feature extractors. We test two paired with the SSD: The first is Inception V2, the ILSVRC 2014 classification and detection winner, which utilizes modules that concatenate efficient decomposed filters [15]. The second is MobileNet, which is optimized for computational efficiency with filters that are further decomposed [14]. With both the Faster R-CNN and R-FCN meta-architecture, we use the ResNet-101 feature extractor, which won the ILSVRC 2015 and COCO 2015 classification and detection and uses residual connections to train very deep networks [17]. On the Faster R-CNN meta-architecture, Inception ResNet v2 enhances the Inception modules with residual connections and *à trous* convolution [19]. MS-CNN's novelty lies in its ability to generate region proposals with multiple scales [11].

While most CNN architectures are limited by the assumption that geometric transformations are fixed and known, Deformable Convnets allow free-form deformation of the sampling grid [20]. Since these deformations are learned from features in upstream layers, they can adapt to the training data. The deformability is also present in the ROI pooling layer, enabling flexible, non-uniform boundaries when performing region proposals. This enhances localization, especially for non-rigid objects.

III. MUTATORS

We call generators of image perturbations "mutators" and divide them into two classes: "Simple" mutators require only the original monocular image. "Contextual" mutators incorporate additional information. 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, which use both to produce simultaneous estimates of scene geometry and optic flow.

A. Simple Mutators

1) Gaussian Blur: Blur the image using a symmetric Gaussian kernel with standard deviations of $\sigma = \{0.5, 1, 1.5, 2, 2.5\}$ pixels. This can be seen as a simple approximation of effects like defocus or material on the lens.

2) Brightness Shift: Apply a multiplicative scaling to each image channel, with saturation, using scale factors of $b = \{0.5, 0.75, 0.875, 1.143, 1.333, 2\}$. This could approximate effects from fast changes in lighting or poor auto-exposure.

3) Alpha Blend: Alpha blend the image with a uniform color, using blending coefficients of $\alpha = \{0.5, 0.75, 0.875\}$. We use a gray color similar to fog, $h = [205, 208, 211]$, to provide a simple approximation of haze effects.

4) JPEG Compression: Perform standard JPEG compression as implemented by the Python Imaging Library. Compression is sometimes applied at image acquisition or during transmission between system components. We used compression quality values of $q = \{60, 40, 20, 10\}$.

5) Salt and Pepper Noise: Some image pixels are selected at random and set to either full black or white, following a commonly-used model of errors in bit transmission or analog-to-digital conversion. The percentage of affected pixels was set to $s = \{1\%, 2\%, 5\%, 10\%\}$.

6) 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.

7) Additive Signal-Dependent Gaussian Noise: Gaussian noise is added per pixel with dependence on the input, simulate camera noises, such as thermal noise, film-grain noise and multiplicative speckle noise [21]:
$P + P^{\psi} \cdot N(0, \zeta_u^2) + N(0, \zeta_w^2)$.

B. Contextual Mutators

To estimate depth throughout the image, we apply PRSM [22], the leading method with a public implementation on the KITTI Scene Flow Benchmark [23], to every test set image to provide context for these mutators. We seed the estimation with some prior knowledge of the vehicle hood location (which will not move) and rectification artifacts (considered planes at infinity).[1] We then refine the estimated depths using a bilateral solver [24] 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 done once.

1) Haze: We use a uniform haze model that has been widely applied and shown to be effective in haze removal [25], which models haze as an alpha-blend effect, whose alpha term increases with distance from the camera:

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

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

$\mathbf{H}(x)$ is an image under the effects of haze. $\mathbf{I}(x)$ represents the scene radiance, in our case the original image. 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, $\mathbf{D}(x)$ is smoothed 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 III-A.3 and density/visibility values shown in Table II. 100m is the lowest visibility distance reported by NOAA before rounding to "zero" [26], so that is the most extreme condition we simulate. Sample images of the result are shown in Figure 2.

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

Mutator	Variable	Values		
Haze	β	0.04	0.012	0.004
	u_V	97.8m	326m	978m
Defocus	κ	2.0	2.8	3.6
	u_f	1m	2m	5m

TABLE II: Parameters used for haze and defocus mutators. Haze scattering coefficients, β, are shown with resulting visibility distances, u_V. For the defocus mutator, a test was run for every combination of focus distance, u_f, and camera constant, κ, values.

Parameter	Symbol	Extreme Value
Aperture f-Number	N	1.4
Pixel Width	γ	1.24×10^{-6} meters
Lens Focal Length	f	2.5×10^{-3} meters

TABLE III: Physical camera parameters used to compute most extreme camera constant used in this work.

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 that pixel normally images. To mutate an image to exhibit defocus, we first compute this 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.

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

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 [27], $\rho(y)$, 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:

$$\rho(x) = \kappa \frac{|\mathbf{D}(x) - u_f|}{\mathbf{D}(x) u_f} \quad (4)$$

We evaluated the effects of three different values of κ for each focal distance, shown in Table II, to consider a range of possible cameras. Example resulting images are shown in Figure 3. The camera constant is the combination of three physical parameters: $\kappa = \frac{\gamma f^2}{N}$ We chose extreme values that could be used in robotics for each parameter to compute our most extreme camera constant. These parameters and their extreme values can be found in Table III.

IV. RESULTS

To establish a baseline, we begin by computing the performance of all SUTs on the unmodified test set. Then we apply each mutator to evaluate robustness to its effects. In all cases, ADR is computed as in Section II-A. This baseline

(a) Original Image, **I** (b) Gaussian Blur, $\sigma = 2$ (c) Brightness Shift, $b = 0.5$ (d) Alpha Blend, $\alpha = 0.5$

(e) JPEG Compression, $q = 40\%$ (f) Salt and Pepper, $s = 10\%$ (g) Channel Drop-Out, $Y(Cb)Cr$ (h) Additive Gaussian Noise, $\sigma_w = 15.0$, $\sigma_u = 0.5$, $\gamma = 0.5$

Fig. 1: Simple mutators applied to a sample image from the Agricultural Person Detection Dataset

$u_V = 3260$m $u_V = 978$m $u_V = 326$m $u_V = 97.8$m

Fig. 2: Example images of different magnitudes of artificial haze applied to the same image as in Figure 1. Choice of visibility distances is described in Section III-B.1.

$u_f = 1$m, $\kappa = 2$ $u_f = 2$m, $\kappa = 2$ $u_f = 2$m, $\kappa = 3.6$ $u_f = 5$m, $\kappa = 3.6$

Fig. 3: Example images of artificial defocus on the same image as Fig. 1. 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.

performance is shown in Figure 4. Ranking the SUTs by ADR puts Deformable Faster R-CNN in the lead overall, and of the two fast methods, SSD Mobilenet does best.[2] On each curve, we have marked the choice of sensitivity that would result in an average FP rate of one every 10 images (triangle), every 100 images (star) and every 1000 images (circle). In the ROCs that follow under mutation, these symbols correspond to the same sensitivities, based on baseline conditions, as described in Section II-A.

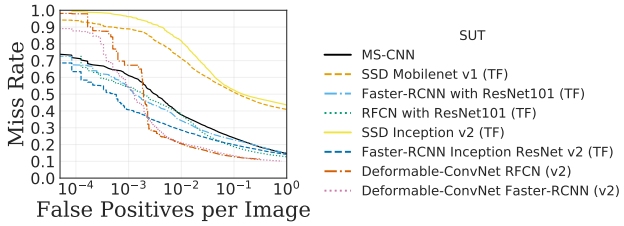
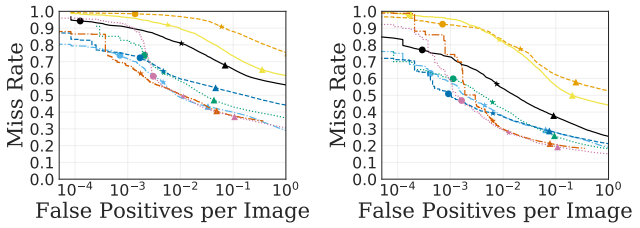


Fig. 4: Baseline test set performance for all SUTs

[2]Although SSD performance is markedly lower than other meta-architectures', it may be sufficient for some applications. Fast methods are most readily adaptable to deployment on embedded systems, and shorter latencies can sometimes lower accuracy requirements.

A. Performance Under Mutations

Behavior of all SUTs under each mutation is shown in Table IV in terms of both absolute detection rates and



(a) $u_f = 1.0$m (b) $u_f = 2.0$m

Fig. 5: Performance under the effects of defocus for $\kappa = 3.6$. See the legend in Figure 4

normalized with respect to baseline performance. No single SUT performs best under all conditions.

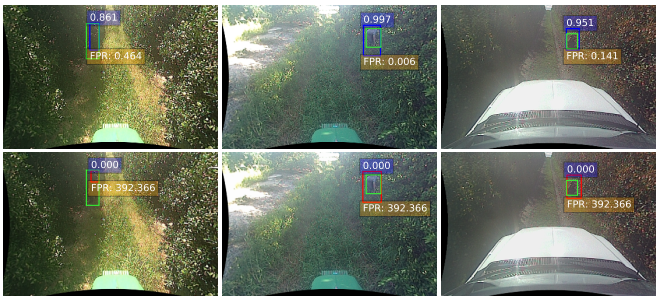
Deformable Faster R-CNN shows good robustness to mutations in addition to strong baseline performance. It maintains its lead under all conditions except channel dropout. If Deformable Faster R-CNN is removed, the second best performer is less clear and would depend more on which of these conditions is most relevant to the domain of application. Although Faster R-CNN with Inception ResNet v2 has the second highest baseline score, it is overtaken by Faster R-CNN with ResNet-101 in several conditions.

In addition to their lower baseline scores, SSD detectors degrade more substantially than other detectors under most mutations, particularly for more extreme mutations. This is most noticeable in channel dropout, which devastates their performance, and at higher levels of alpha blending.

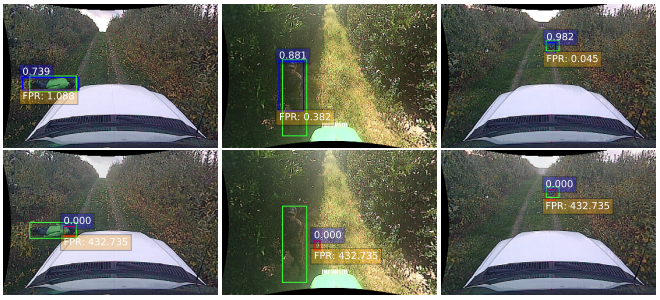
Performance under defocus conditions shows similar trends to that under Gaussian blur; SUTs that are robust to one tend also to be robust to the other. The trend is similar for haze and alpha blend, but with some outliers. These relationships are explored further in Section IV-B. Full ROCs are shown for defocus in Figure 5, which can be compared to Figure 4 to see changes. Note that mutations affect both miss rate and FP rate and that for some SUTs the overall shape of the curve changes. For these SUTs, the full ROC provides more insight than the aggregate ADR.

JPEG compression has no appreciable effect on performance until JPEG qualities below 40%, but does not show major degradation in performance until a level of 10%, when artifacts are clearly visible to humans. Brightness changes have very little effect on most SUTs, but brightening images significantly hurts the SSD SUTs. However, the same brightening actually increases MS-CNN performance above baseline. Most SUTs are robust to both salt and pepper and additive noise until they reach the highest magnitudes. The SSD-based SUTs see degradations at earlier levels though. Channel dropout results in the biggest drops in performance from the largest share of SUTs. Only the TensorFlow Faster R-CNN-based SUTs maintain reasonable performance under that mutation, and the SSD detectors drop to producing almost no correct detections.

Even in the case of mutations that show relatively small changes in ADR, the effects on individual images can be striking. Figure 6 illustrates how imperceptible changes can



(a) Blur



(b) Haze

Fig. 6: Examples of images that show the largest change in detection performance for MS-CNN under moderate blur and haze. For all of them, the rate of FPs per image required to detect the person increases by three to five orders of magnitude. In each image, the green box shows the labeled location of the person. The blue and red boxes are the detection produced by the SUT before and after mutation respectively, and the white-on-blue text is the strength of that detection (ranged 0 to 1). Finally, the value in white-on-yellow text shows the average FP rate per image that a sensitivity threshold set at that value would yield. i.e., that is the required FP rate to still detect the person.

sometimes result in dramatically different behavior. Note that these perturbations are not adversarial in nature, but over prolonged operation they can still happen by chance.

B. Predicting Contextual Performance

The contextual mutators evaluated in this work, defocus and haze, incorporate context in the form of scene depths, but otherwise apply the same transformations as simpler mutators (blur and alpha blend respectively), with magnitude dependent on scene depth. A natural question is whether this context is important to the evaluation. One crucial use is to relate magnitudes of blurring or blending to real, physical phenomena, for imposing deployment requirements on a system. Additionally, there may be important interactions from the effects of multiple depths. To evaluate this, we attempted to predict the effects of contextual mutators from the performances on the corresponding simple mutators.

We perform prediction of ADRs (normalized to focus on robustness, rather than gross performance) for each contextual mutator configuration independently, and for each SUT by using a leave-one-SUT-out scheme. Data for all simple mutators for all but one SUT are used to fit a non-negative

Mutator & Parameters	MS-CNN	SSD w/ MobileNets	SSD w/ Inception	Faster R-CNN w/ Resnet 101	R-FCN w/ Resnet 101	Faster R-CNN w/ Inception Resnet	Deformable R-FCN	Deformable Faster R-CNN
Baseline	**0.60**	**0.29**	**0.22**	**0.64**	**0.64**	**0.71**	**0.71**	**0.73**
Defocus (u_f 10.0; κ 2.0)	0.59	0.29	0.22	0.64	0.63	0.71	0.63	0.73
Defocus (u_f 5.0; κ 2.0)	0.59	0.29	0.22	0.64	0.64	0.71	0.63	0.73
Defocus (u_f 2.0; κ 2.0)	0.52	0.29	0.24	0.63	0.63	0.68	0.62	0.74
Defocus (u_f 1.0; κ 2.0)	0.38	0.20	0.21	0.54	0.53	0.57	0.50	0.69
Defocus (u_f 10; κ 2.8)	0.59	0.29	0.22	0.64	0.63	0.71	0.63	0.73
Defocus (u_f 5; κ 2.8)	0.59	0.29	0.23	0.64	0.64	0.71	0.63	0.73
Defocus (u_f 2; κ 2.8)	0.47	0.26	0.23	0.59	0.59	0.65	0.58	0.73
Defocus (u_f 1.0; κ 2.8)	0.27	0.14	0.17	0.47	0.44	0.44	0.40	0.58
Defocus (u_f 10.0; κ 3.6)	0.59	0.29	0.22	0.64	0.63	0.71	0.63	0.73
Defocus (u_f 5.0; κ 3.6)	0.57	0.29	0.23	0.64	0.63	0.70	0.62	0.73
Defocus (u_f 2.0; κ 3.6)	0.43	0.24	0.22	0.55	0.56	0.60	0.53	0.70
Defocus (u_f 1.0; κ 3.6)	0.19	0.11	0.13	0.42	0.38	0.36	0.34	0.51
Gaussian Blur (σ 0.5)	0.56	0.29	0.23	0.64	0.64	0.70	0.63	0.74
Gaussian Blur (σ 1.0)	0.48	0.27	0.24	0.61	0.61	0.67	0.60	0.74
Gaussian Blur (σ 1.5)	0.41	0.22	0.22	0.56	0.56	0.61	0.54	0.71
Gaussian Blur (σ 2.0)	0.33	0.17	0.19	0.51	0.49	0.53	0.47	0.65
Gaussian Blur (σ 2.5)	0.25	0.13	0.16	0.47	0.44	0.45	0.41	0.59
Gaussian Blur (σ 3.0)	0.19	0.10	0.14	0.43	0.40	0.37	0.35	0.53
Haze (u_V 978.0 m (β 0.004))	0.56	0.29	0.22	0.64	0.64	0.69	0.63	0.73
Haze (u_V 326.0 m (β 0.012))	0.50	0.28	0.21	0.64	0.65	0.67	0.63	0.73
Haze (u_V 97.8 m (β 0.04))	0.36	0.19	0.14	0.61	0.60	0.61	0.61	0.71
Alpha Blend (α 0.1)	0.53	0.29	0.21	0.64	0.64	0.69	0.63	0.73
Alpha Blend (α 0.25)	0.38	0.24	0.18	0.64	0.62	0.66	0.63	0.73
Alpha Blend (α 0.5)	0.22	0.05	0.09	0.63	0.55	0.63	0.63	0.72
Alpha Blend (α 0.75)	0.21	0.00	0.00	0.54	0.28	0.55	0.59	0.67
JPEG Compression (q 40)	0.56	0.27	0.21	0.62	0.61	0.68	0.61	0.71
JPEG Compression (q 20)	0.51	0.25	0.19	0.57	0.57	0.64	0.58	0.68
JPEG Compression (q 10)	0.39	0.19	0.15	0.47	0.46	0.51	0.49	0.58
Brightness (b 2.00)	0.61	0.14	0.09	0.51	0.59	0.60	0.59	0.66
Brightness (b 1.33)	0.63	0.25	0.16	0.60	0.64	0.66	0.63	0.72
Brightness (b 1.14)	0.61	0.27	0.19	0.62	0.64	0.69	0.63	0.73
Brightness (b 0.88)	0.57	0.30	0.25	0.65	0.63	0.72	0.62	0.73
Brightness (b 0.75)	0.55	0.30	0.26	0.64	0.62	0.73	0.62	0.72
Brightness (b 0.50)	0.56	0.24	0.23	0.61	0.58	0.73	0.60	0.71
Salt and Pepper (1% of pixels)	0.58	0.27	0.20	0.60	0.61	0.66	0.61	0.70
Salt and Pepper (2% of pixels)	0.55	0.25	0.18	0.57	0.59	0.63	0.60	0.68
Salt and Pepper (5% of pixels)	0.50	0.21	0.14	0.51	0.54	0.58	0.55	0.61
Drop Channel Cb (YCbCr)	0.36	0.01	0.00	0.40	0.09	0.41	0.16	0.11
Drop Channel Cr (YCbCr)	0.30	0.00	0.00	0.33	0.04	0.49	0.13	0.10
Drop Channel R (RGB)	0.64	0.07	0.01	0.51	0.34	0.56	0.34	0.37
Drop Channel G (RGB)	0.49	0.03	0.00	0.45	0.23	0.60	0.28	0.32
Drop Channel B (RGB)	0.40	0.03	0.03	0.39	0.23	0.58	0.29	0.29
Additive (ζ_w 5.0; ζ_u 0.5; ψ 0.5)	0.60	0.28	0.21	0.63	0.62	0.69	0.62	0.71
Additive (ζ_w 5.0; ζ_u 0.5; ψ 0.7)	0.60	0.27	0.19	0.61	0.60	0.66	0.60	0.68
Additive (ζ_w 5.0; ζ_u 1.5; ψ 0.5)	0.60	0.26	0.19	0.61	0.59	0.65	0.59	0.66
Additive (ζ_w 15.0; ζ_u 0.5; ψ 0.5)	0.59	0.25	0.18	0.60	0.58	0.65	0.59	0.66
Additive (ζ_w 5.0; ζ_u 2.5; ψ 0.5)	0.59	0.21	0.15	0.56	0.54	0.60	0.55	0.60

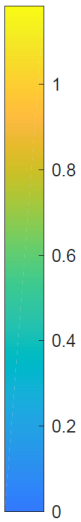
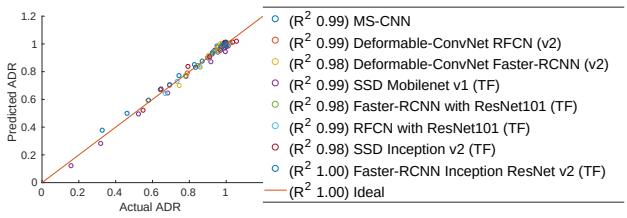


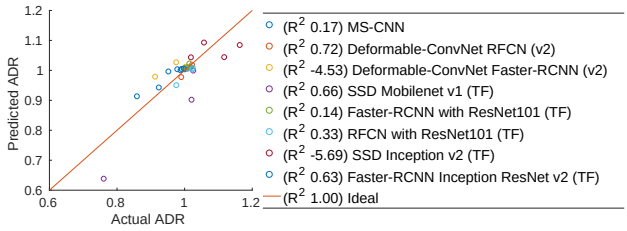
TABLE IV: ADRs for each SUT under all mutations. Numerical values show ADR, while cell colorization depicts ADR normalized relative to that SUT's baseline score, to highlight robustness characteristics. The color bar at right shows the normalized scale's color mapping; note that performance can sometimes improve over baseline.

least-squares model to predict that SUT's performance on a given contextual mutator parameterization. Intuitively, this emulates trying to leverage knowledge of the robustness relationships between simple and contextual mutators to estimate performance without contextual information.

Figure 7 shows a comparison of the predicted and observed normalized detection rates for each mutator. The blur prediction model approximates defocus very well, giving R^2 values ≥ 0.98 for all SUTs. Prediction haze performance is more tenuous though, with poor predictions of several SUTs. Figure 8 visualizes the models fit for both mutators. In each plot, blue bars show the contribution of that simple mutator to predicting the corresponding contextual mutator. Yellow bars show a histogram of the amount of blurring or blending applied to the labeled region of the person, throughout the dataset, for each contextual mutator configuration. These are well-correlated for defocus, suggesting that defocus effects could be estimated well from blur for cases where the depths of the important regions of the image are well-understood. The fit is fairly well-correlated with this histogram for haze as well, despite its poor performance at prediction, suggesting there are important interactions that are not well-captured by considering depths independently.

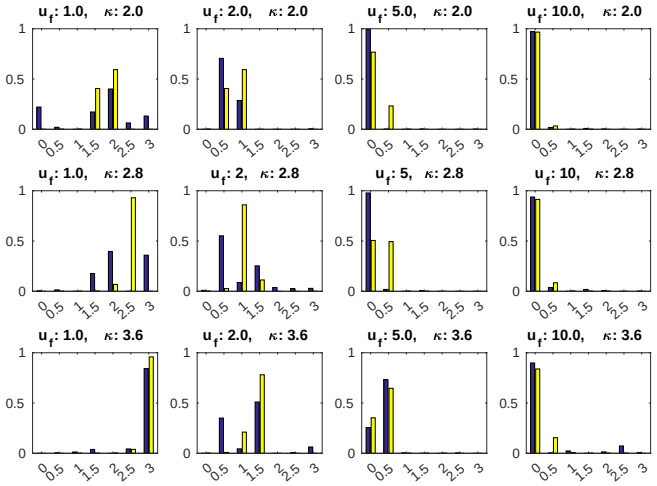


(a) Defocus as predicted by blur

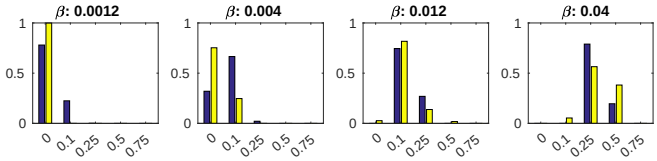


(b) Haze as predicted by alpha blend

Fig. 7: Regression results for predicting contextual mutator performances from those of simple mutators



(a) Defocus



(b) Haze

Fig. 8: Distribution fits for predicting performance on defocus from blur and haze from alpha blend (blue), compared to the distributions of simple mutator magnitudes (yellow)–blur for defocus and alpha blend for haze respectively–resulting within person label bounding boxes when contextual mutators are applied to the dataset.

C. Threats to Validity

1) Generalization outside our chosen mutators: We chose a set of mutators, which can not cover the full range of difficult cases that a vision system will be exposed to [28]. For our contextual metrics, we chose common approximations, but these approximations may not fully capture relevant physical effects. The mutators we have chosen include several where high frequency information is suppressed in the image. Our conclusions may not generalize to other difficult cases, such as dirty lens or glare effects. We do include some simple mutators that add information, including salt and pepper and additive Gaussian noise. Testing more additive mutators will be explored in future work.

2) Generalization outside our chosen SUTs: We have chosen a number of object detectors to test, but these may not be representative of other current or future detection systems. Our focus is more on the process than the individual SUTs though, and we have tried to choose algorithms that have a range of inference times and are best in class on a variety of datasets. These detectors are tied to the method used to train them, which varies by implementation and base library. While it's impractical to try every training configuration, we strove to make the process consistent, to avoid, for instance, some SUTs training on the mutations.

3) Generalization outside our dataset: Our chosen benchmark problem also imposes some assumptions. We use a modified evaluation metric [10], which may not capture the true performance relative to safety objectives. We have used the largest scale person dataset available with all the required secondary information. We have so far only evaluated this work on a single off-road person detection dataset; it may not extend to urban environments, which we plan to evaluate in future work. Finally, these conclusions may not be applicable to other safety-critical computer vision problems, such as odometry or scene understanding. We hope that as autonomous vehicles get closer to application, we and others will take a critical look at every step in the process.

V. CONCLUSIONS

We present a method for evaluating the robustness of image-based algorithms to a variety of image modifications, to allow estimation of how they would respond to unexpected or uncommon conditions without requiring exhaustive real-world testing under every condition. We show that the choice of best-performing algorithm for a robotic system may change when robustness is considered, and that choice will depend on the conditions relevant to the application.

We also introduce two mutations that make use of additional context, in the form of scene depths, to perform more physically-realistic simulation of the effects of defocus and haze. For both the simple and contextual mutators, we demonstrate cases where performance drops catastrophically in the presence of barely-perceptible changes.

Each of the contextual mutators consist of a depth-varying application of one of the simple mutators that does not require context, and we show the relationship between robustness to the simple mutators and to the contextual

mutator. In the case of defocus, we show promising results for predicting performance under the more physically realistic contextual mutator, even without needing that image context. From evaluation on a range of different mutations, generalized robustness trends begin to emerge that hint at likely robustness to other unseen effects.

ACKNOWLEDGMENT

This project was funded by the Test Resource Management Center (TRMC) 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 2018-165. Distribution Statement A - "Approved for public release; distribution is unlimited".

References

[1] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" *Transportation Research Part A: Policy and Practice*, vol. 94, pp. 182–193, 2016.

[2] P. Koopman and M. Wagner, "Challenges in autonomous vehicle testing and validation," *SAE International Journal of Transportation Safety*, vol. 4, no. 2016-01-0128, pp. 15–24, 2016.

[3] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *CVPR*, 2016.

[4] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, K. Rosaen, and R. Vasudevan, "Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks?" In *ICRA*, 2017.

[5] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[6] A. Athalye and I. Sutskever, "Synthesizing robust adversarial examples," *arXiv preprint arXiv:1707.07397*, 2017.

[7] S. Dodge and L. Karam, "Understanding how image quality affects deep neural networks," in *Quality of Multimedia Experience (QoMEX)*, 2016.

[8] S. Karahan, M. K. Yildirum, K. Kirtac, F. S. Rende, G. Butun, and H. K. Ekenel, "How image degradations affect deep cnn-based face recognition?" In *BIOSIG*, 2016.

[9] B. Richard-Webster, S. E. Anthony, and W. J. Scheirer, "Psyphy: A psychophysics driven evaluation framework for visual recognition," *arXiv preprint arXiv:1611.06448*, 2016.

[10] Z. Pezzementi, T. Tabor, P. Hu, J. K. Chang, D. Ramanan, C. Wellington, B. P. W. Babu, and H. Herman, "Comparing apples and oranges: Off-road pedestrian detection on the national robotics engineering center agricultural person-detection dataset," *Journal of Field Robotics*, 2017.

[11] Z. Cai, Q. Fan, R. Feris, and N. Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection," in *ECCV*, 2016.

[12] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," *arXiv preprint arXiv:1611.10012*, 2016.

[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *ECCV*, 2016.

[14] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[15] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *CVPR*, 2016.

[16] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *NIPS*, 2016.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[18] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *NIPS*, 2015.

[19] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning.," in *AAAI*, 2017.

[20] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," *arXiv preprint arXiv:1703.06211*, 2017.

[21] X. Liu, M. Tanaka, and M. Okutomi, "Estimation of signal dependent noise parameters from a single image," in *ICIP*, 2013.

[22] C. Vogel, K. Schindler, and S. Roth, "3d scene flow estimation with a piecewise rigid scene model," *IJCV*, vol. 115, no. 1, pp. 1–28, 2015.

[23] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *CVPR*, 2015.

[24] J. T. Barron and B. Poole, "The fast bilateral solver," in *ECCV*, 2016.

[25] K. He, J. Sun, and X. Tang, "Single image haze removal using dark channel prior," *PAMI*, vol. 33, no. 12, pp. 2341–2353, 2011.

[26] Visibility, *Visibility — Wikipedia, the free encyclopedia*, [Online; accessed 7-February-2018], 2018. [Online]. Available: `https://en.wikipedia.org/wiki/Visibility%20&oldid=819466260`.

[27] S. D. P. Arroyo, "Modeling and applications of the focus cue in conventional digital cameras," PhD thesis, Universitat Rovira i Virgili, 2013.

[28] O. Zendel, M. Murschitz, M. Humenberger, and W. Herzner, "Cv-hazop: Introducing test data validation for computer vision," in *ICCV*, 2015.