Monitoring and Adapting the Physical State of a Camera for Autonomous Vehicles

Maik Wischow[1,2], Guillermo Gallego[2], Ines Ernst[1], Anko Börner[1]

Abstract—Autonomous vehicles and robots require increasingly more robustness and reliability to meet the demands of modern tasks. These requirements specially apply to cameras onboard such vehicles because they are the predominant sensors to acquire information about the environment and support actions. Cameras must maintain proper functionality and take automatic countermeasures if necessary. Existing solutions are typically tailored to specific problems or detached from the downstream computer vision tasks of the machines, which, however, determine the requirements on the quality of the produced camera images. We propose a generic and task-oriented self-health-maintenance framework for cameras based on data- and physically-grounded models. To this end, we determine two reliable, real-time capable estimators for typical image effects of a camera in poor condition (blur, noise phenomena and most common combinations) by evaluating traditional and customized machine learning-based approaches in extensive experiments. Furthermore, we implement the framework on a real-world ground vehicle and demonstrate how a camera can adjust its parameters to counter an identified poor condition to achieve optimal application capability based on experimental (non-linear and non-monotonic) input-output performance curves. Object detection is chosen as target application, and the image effects motion blur and sensor noise as conditioning examples. Our framework not only provides a practical ready-to-use solution to monitor and maintain the health of cameras, but can also serve as a basis for extensions to tackle more sophisticated problems that combine additional data sources (e.g., sensor or environment parameters) empirically in order to attain fully reliable and robust machines.

Index Terms—Autonomous robots, robot control, smart cameras, image quality, deep learning, object detection.

I. INTRODUCTION

MACHINES from different fields (e.g., vehicles, robots) are indispensable to facilitate and automate tedious, time-consuming or hazardous tasks. As a result, there is a strong incentive to continually advance them away from manual control towards greater levels of autonomy. This increasing autonomy enables several new application areas, such as mapping unknown environments in exploration missions [1], navigation in search and rescue operations [2], parking surveillance [3], or delivering tasks in manufacturing and warehousing [4]. However, in all these use cases, machines are susceptible to a variety of dynamic environmental factors (object motion [5], lighting [6], temperature [7], weather [8],

etc.). These factors have a direct impact on how their sensors perceive the environment, which, in turn, affects their subsequent actions [9]–[11]. Hence, to ensure the safety of both individuals and the machines themselves, special attention must be paid to reliable and robust sensors. Cameras are nowadays the predominant sensors to perceive the environment, and are therefore the subject of this study. To guarantee a camera's intended functionality, autonomy demands for self-health-maintenance, i.e., the task of continuously monitoring the behavior of the system and executing automatic countermeasures in case of a detected misbehavior.

Previous studies (e.g., [12]–[16]) have approached this task by monitoring and optimizing image features linked to general image quality (like sharpness, noise or dynamic range). To this end, various automatic image quality maintenance techniques have been developed and are now part of a standard camera's imaging pipeline (auto-focus, auto-exposure, auto-calibration, etc.). However, these techniques are typically decoupled from the downstream vision application (e.g., environment mapping, object detection, or navigation) and hence may not reach optimal application performance. This is particularly true if the system can trade off image quality for other vision application benefits. Moreover, each application has its own requirements for what is considered an optimal image quality.

This work closes the gap by proposing a general self-health-maintenance framework that strives for optimal application performance. The framework combines three key components: a continuous image quality monitoring, knowledge about the requirements of a downstream vision application, and a camera control for precise image quality adjustments. We demonstrate the working principle of our framework on the exemplary application of object detection (as a representative modern vision application of great importance in various fields), and focus on motion blur and noise as typical undesired image properties (see Fig. 1). Our modular design favors interpretability, explainability, and testability of individual components compared to end-to-end approaches. Without loss of generality, our study analyzes: time-varying effects influencing blur and noise quality parameters (since any time-invariant effects are usually subtracted by camera calibration), and region-wise effects, thus allowing us to consider spatially-varying problems.

We make the following contributions:

- We propose a general framework to approach camera self-health-maintenance by maximizing the performance of arbitrary downstream vision applications through continuous monitoring and adjustment of image quality.
- We demonstrate our framework running in real-time on a real-world ground vehicle for the application of object

[1]M.W., I.E. and A.B. are with the German Aerospace Center (DLR), Berlin, Germany. E-Mail: [firstname].[lastname]@dlr.de.

[2]G.G. is with the Dept. of EECS of TU Berlin (Faculty IV), the Einstein Center Digital Future, and the Science of Intelligence Excellence Cluster, Berlin, Germany. E-Mail: guillermo.gallego@tu-berlin.de.
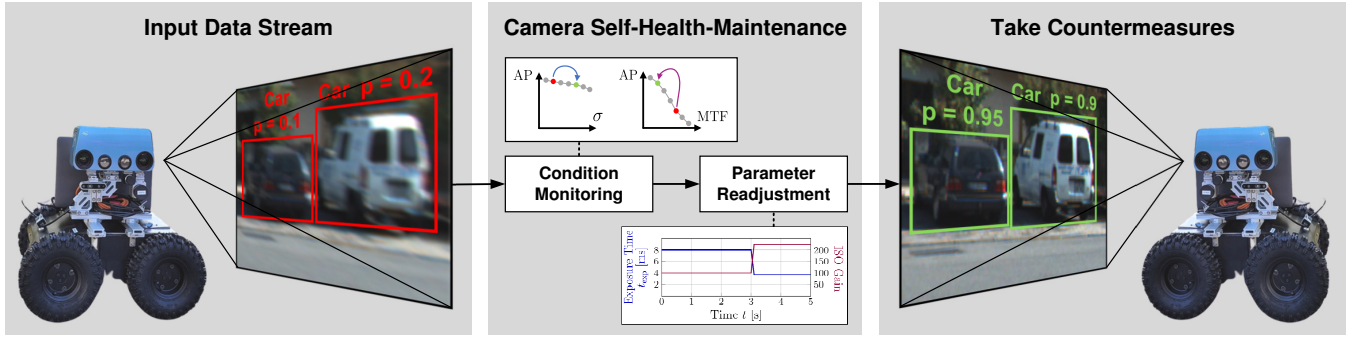
Fig. 1: The ground vehicle fails to detect the motion blurred cars (red) given its current camera configuration. We tackle the source of this problem using (i) an online estimation of image quality properties, (ii) knowledge about camera physics and (iii) empirical object detection performance curves AP (expressed as functions of the image quality). In this way, unfavorable camera conditions can be detected and actively tackled to reach optimal application performance (green). In the example, image blur is estimated and mitigated online by changing the camera configuration: decreasing the exposure time t_{exp} and increasing the ISO gain. Blur is reduced at the expense of slightly increasing noise to produce better object detection rates.

detection (e.g., "car", "pedestrian") affected by motion blur and noise.

- We evaluate two customized machine-learning (ML) based image blur and noise estimators in an experimental study: we compare them to four traditional state-of-the-art estimators on three datasets, account for five isolated and two combined blur and noise root causes grounded in knowledge of camera physics, and propose a post-processing step to re-enable blur estimation in presence of high noise.
- Our experiments yield practical recommendations for the robustness of camera monitoring applications.
- We provide the source code of our experiments and of all estimators: https://github.com/MaikWischow/Camera-Condition-Monitoring.

II. RELATED WORK

Our study is closely related to active vision [17] and adaptive camera regulation [18] in that there are two connected tasks: online *estimation* of the current vision state and execution of an *action* to improve some target criterion. In the estimation task, we estimate major properties of the camera system state by assessing the quality of the image data *it produces* in terms of blur and noise. Subsequently, we define actions that can be carried out to control the camera, therefore influence image properties (we demonstrate this for motion blur and noise) and hence optimize the system's performance for a target application (object detection in this work).

Motion blur can be directly approached at a hardware level by involving, e.g., an accelerometer [19] or a self-designed sensor [20], but it is typically managed by automatic exposure control through image processing. Most image-based algorithms represent optimal exposure selection as a control problem on image quality indicators like the intensity entropy [12], [14], gradients [13], [14] and histograms [21], [22], or approach it learning-based [23]. Our work is most similar to [14] and [23], thus we use both as comparison baselines.

The study of Shin et al. [14] is likewise motivated by challenging image effects such as motion blur or noise, which

"can be dramatically alleviated by carefully adjusting camera exposure parameters". To this end, the authors propose an exposure time and gain control that maximizes image entropy, gradient strength, and gradient uniformity while minimizing noise. However, this procedure does not account for the downstream vision applications that determine the required image quality (as most traditional approaches, such as [12], [13], [22], [24]). Furthermore, they assume simplified additive Gaussian noise. In contrast, we consider an extensive image formation pipeline that models the most important noise sources close to the camera physics (which is experimentally supported as being more realistic [25], [26]). We also include object detection performance as a feedback signal and hence aim for optimal downstream application performance.

Onzon et al. [23] propose a neural network for auto-exposure control that is trained jointly, end-to-end with an object detector. Unlike [14], the authors consider an extensive noise formation pipeline. On top of this, we assume a more comprehensive and realistic image formation by additionally including motion and defocus blur as well as all image corruptions occurring simultaneously and influencing each other (cf. [27]). Moreover, we do not rely on a tailored end-to-end learning approach, but instead propose a more modular, extensible and interpretable concept: given the image data, we empirically determine a performance profile (adapted to the application) in terms of data quality metrics. To this end, we employ dedicated blur and noise estimators with real-time capabilities by adapting [13], [14] and [23] to focus on regions of interest. The details are presented in the upcoming section.

Recent work also investigates environmental impacts (e.g., thermo-mechanical stress or mechanical vibrations) on defocus blur, resolving power, or camera calibration [7], [24]. While these are based on fundamental studies, we focus on approaches directly applicable in practice.

III. PROPOSED SYSTEM

We introduce the proposed system in a top-down approach. First, an overview is provided (Sec. III-A). Next, we explain its underlying working principle, which connects camera physics,

blur, noise, and object detection performance (Sec. III-B). We then detail the framework's components. We start with the creation of an application performance profile using the example of object detection (Sec. III-C). Finally, we present employed traditional and learning-based (ML) blur and noise estimation methods to quantify image quality objectively, and address the necessary changes we made (Secs. III-D and III-E).

A. Overview

Our proposed camera self-health-maintenance system consists of online testing (Fig. 2) and offline training parts (Fig. 3).

Let us briefly introduce the offline training procedure first, as training happens before the testing/inference phase. We start with image datasets from a target application domain as input (e.g., object detection) and corrupt them according to an image formation pipeline (Fig. 4). The pipeline contains the most common (physics-based) sources of blur and noise affecting the camera condition, with realistic severity levels (Sec. IV-A). We quantify these levels using objective noise and blur metrics: noise level σ and modulation transfer function (MTF) values, respectively. Afterwards, we let our system's target application (object detection) evaluate these corrupted images. We likewise quantify this performance in terms of the well known average precision score (AP, Sec. III-C). Knowing each applied image corruption and the corresponding calculated application performance, the respective tuples are aggregated into input/output performance curves (IOPC), which is the final product of this training procedure.

The testing part (Fig. 2) has access to these IOPCs and analyzes each captured (yet unprocessed) camera image online using ML-based, real-time capable noise level and MTF estimators (Secs. III-D and III-E). We evaluate their estimation and runtime performances compared to established state-of-the-art estimators (Sec. IV) for isolated and combined corruption cases, and propose a simple approach to improve blur estimation in case of interfering high noise levels (Sec. IV-D). If the estimated image quality does not meet the requirements for optimal application performance recorded in an IOPC, a control policy decides how to adjust camera parameters as countermeasure. We propose two exemplary control policies using exposure time and ISO gain to trade off blur and noise. They exploit the fact that object detectors are typically more sensitive to blur than to noise (Sec. V).

B. Optimize Object Detection by Trading off Blur and Noise

We now demonstrate how one can use the online blur/noise estimators and the offline empirical IOPCs to control image quality and hence optimize object detection performance (Fig. 2). Here we focus on actions tackling linear motion blur (Lin. MB) because object detectors are substantially more sensitive to Lin. MB than to noise (Fig. 15), and there is abundant motion blur in standard datasets like Udacity (Fig. 10).

We make the following considerations knowing the camera's physical processes. The main controllable influencing factor of
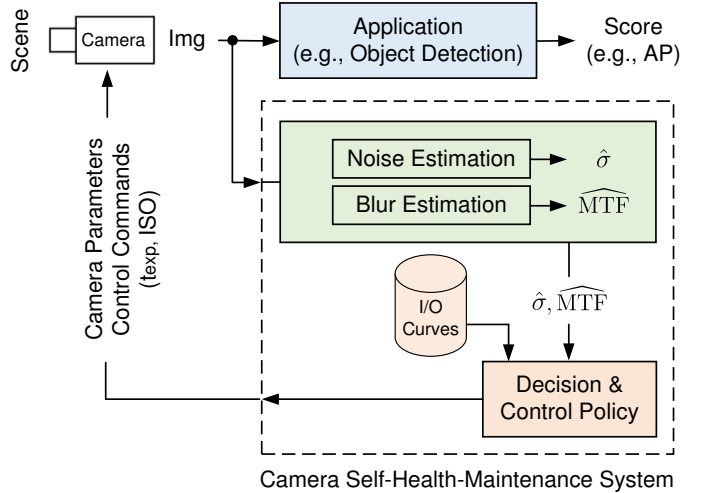

Fig. 2: *System overview*. The camera is constantly monitored by analyzing image corruptions (e.g., blur and noise). According to the estimated severity of such corruptions, camera control parameters (e.g., exposure time t_{exp} and ISO gain) are recalculated to maximize application performance using the (offline determined) input/output (I/O) performance curves.
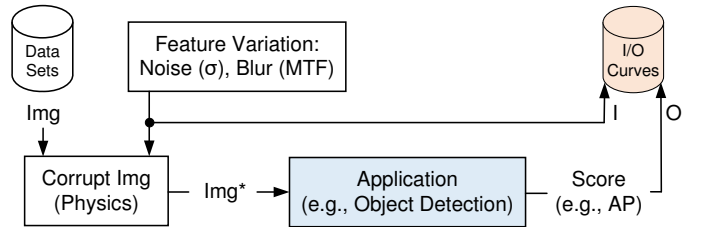

Fig. 3: *Training the system (system identification)*. An offline sensitivity analysis determines the impact of physical image corruptions (e.g., blur and noise) on the performance of a target application (e.g., object detection), and stores the results in input/output (I/O) performance curves. As input, image data close to the application domain are used.

motion blur is the camera's exposure time t_{exp}. We exploit the relations

$$t_{exp} \propto I \text{ and } t_{exp} \propto \text{MB} \sim \text{MTF}^{-1} \sim \text{AP},$$
$$\text{ISO} \propto I \text{ and } \text{ISO} \propto \sigma \sim \text{AP}^{-1}, \quad (1)$$

where AP is the average precision of the object detector.

Changing t_{exp} by a factor $\alpha \doteq t_{exp}^{old}/t_{exp}^{new}$ equally changes the aggregated amount of light intensity I and also the motion blur y the same factor (for simplicity and without loss of generality, we assume linearity of the sensor digitization process [28]). To compensate for the changed light, we may alter the camera ISO gain by factor α, which likewise changes the noise level σ. This relationship depends on the camera sensor architecture and whether the analog or digital signal is amplified [29]. We assume digital amplification as the worst case and thus a linear relation. Hence, we can model the problem as an optimization one, i.e., determining α from the IOPCs to maximize the object detector's score:

$$\alpha^\star = \arg\max_\alpha \text{ AP}(\alpha\,\hat{\sigma}, \alpha\,\text{MB}(\widehat{\text{MTF}})), \quad (2)$$
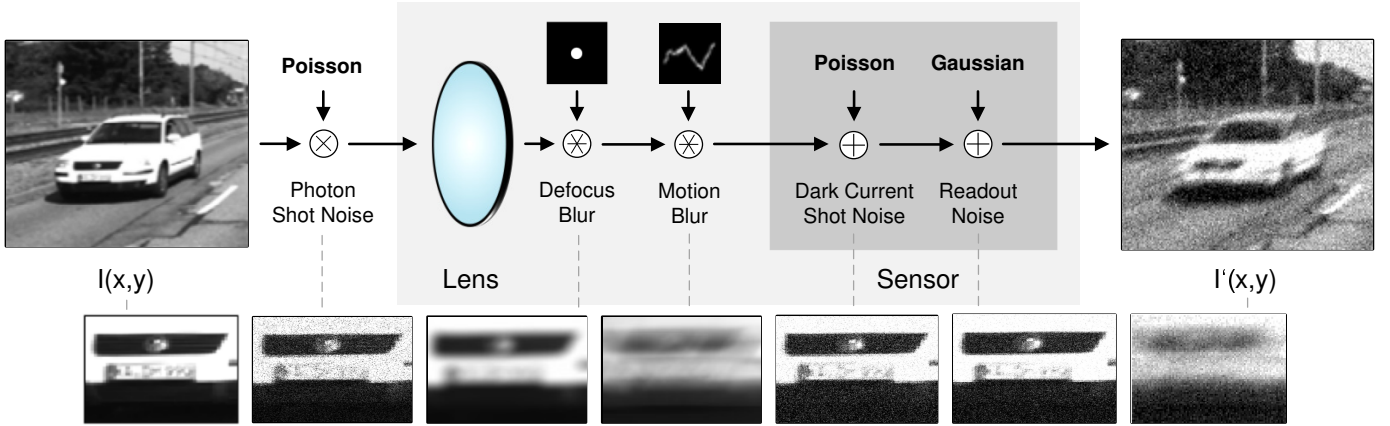

Fig. 4: *Image formation process* of the considered camera system, including blur and noise models. A clean image $I(x,y)$ undergoes several physical processes that produce noise and blur, yielding the corrupted image $I'(x,y)$ (clean image patch vs. distinct corruptions in stated order). Noise is either signal-dependent or signal-independent, while blur is modelled as a convolution with a point spread function (PSF). Details are provided in the supplementary material.
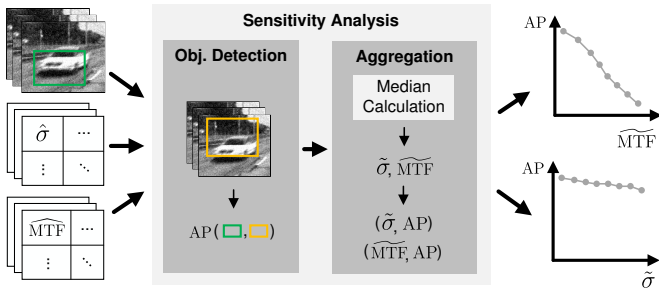

Fig. 5: *Sensitivity analysis* of object detector performance for blur and noise. The detectors are evaluated on corrupted images resulting in average precision (AP) scores. For the true detection areas, corresponding patch-wise noise ($\hat{\sigma}$) and blur ($\widehat{\text{MTF}}$) estimations are aggregated to medians ($\tilde{\sigma}$ and $\widetilde{\text{MTF}}$) and, together with the APs, added to performance curves.

where $\widehat{\text{MTF}}$ and $\hat{\sigma}$ denote the online blur and noise estimations, respectively. Note that $t_{\text{exp}} \propto \sigma_{\text{DCSN}}^2$ during optimization [30, p. 3]. We drop this influence here for simplicity, since the small DCSN has no significant effect on $\hat{\sigma}$ in our setting.

C. Empirical Input-Output Performance Curves

Due to the non-linear and non-monotonic nature of vision applications, such as ML-based object detectors, we aim to determine system output sensitivities *empirically* for different noise and blur levels (Fig. 5). In this work, we use YOLOv4 [31] (YOLOv7 [32] in the supplementary material) and Faster R-CNN [33] as state-of-the-art real-time object detectors (with pre-trained models and default settings, applied on grayscale images). The analysis is performed offline, but an online approach is also feasible.

Let us explain the offline procedure on the example of images with a fixed blur level MTF and noise level σ. As input we assume N_I images $\mathbf{I} \doteq \{I_i(x,y)\}_{i=1}^{N_I}$, where an image I has N_{GT} corresponding ground truth object detections $\mathbf{B}_I^{\text{GT}} \doteq \{b_{I,i}^{\text{GT}}(x,y)\}_{i=1}^{N_{\text{GT}}}$, patch-wise blur estimations

$\widehat{\text{MTF}}_I(x,y)$, and noise estimations $\hat{\sigma}_I(x,y)$; x and y index respective pixel values. A pixel of a $b_I^{\text{GT}}(x,y)$ containing an object is defined as 1 and 0 otherwise.

First, both object detectors are applied to all images $I \in \mathbf{I}$ yielding the estimated object detections $\mathbf{B}_I^{\text{D}}$ per detector and image. Second, these $\mathbf{B}_I^{\text{D}}$ are evaluated against the $\mathbf{B}_I^{\text{GT}}$ using the well-known average precision (AP) metric, which we calculate following [34]. In a subsequent aggregation step, we determine median blur and noise estimations of all image patches overlapping with the ground truth object detections, where

$$\tilde{\sigma} \doteq \text{med}(\{\hat{\sigma}_I(x,y)|b_I^{\text{GT}}(x,y) = 1, \forall I \in \mathbf{I}, \forall x, y \in \mathbb{N}\}) \quad (3)$$

and $\widetilde{\text{MTF}}$ is defined analogously. To bound the complexity, we quantize the estimation parameter spaces into bins with $\tilde{\sigma} \in \{0, 5, \dots, 25\}\,\text{DN}$ and $\widetilde{\text{MTF}} \in \{0.1, 0.2, \dots, 1.0\}$. Finally, the resulting input-output tuples $(\tilde{\sigma}, \text{AP})$, $(\widetilde{\text{MTF}}, \text{AP})$ or $(\tilde{\sigma}, \widetilde{\text{MTF}}, \text{AP})$ are collected as performance curves (IOPCs).

D. Blur Estimation (via the MTF)

The goal of our image blur estimators is to predict the MTF given a possibly blurred input image patch I^*, where I^* is assumed to be monochrome (i.e., grayscale) and of size 192×192 pixels (following the ML approach). Figure 6 summarizes the steps of the two main approaches.

1) Traditional methods (non-learning–based): We use two baseline methods: "graph-based" [35] (Graph-Based Blind Image Deblurring, or simply GBB) and "simple local minimal intensity prior" [36] (Patch-wise Minimal Pixels – PMP) as traditional blur kernel estimators (top branch in Fig. 6). Both estimators follow a maximum-a-posteriori framework

$$\min_{I,h} \mathcal{L}(I \circledast h, I^*) + \alpha G(I) + \beta R(h) \quad (4)$$

to iteratively refine a clean latent image I and the blur kernel h (see supplementary material for details on the image formation process). The objective function (4) is the negative logarithm of the posterior distribution (thus maximization turns
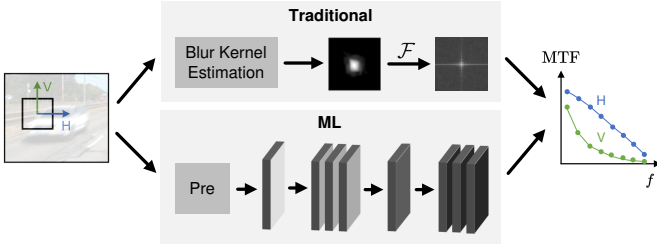
Fig. 6: *Blur estimation* of traditional (top branch) and learning-based (bottom branch, ML) approaches. All methods input one or more image patches and output estimated MTF samples for pre-defined image frequencies (f) in the horizontal (H) and vertical (V) directions. Traditional methods first estimate a blur kernel, transform it into the Fourier space $\mathcal{F}$, and sample MTF values. The learning-based method consists of a pre-processing stage (Pre) followed by a multi-layer CNN.

into minimization). It consists of a data fidelity term $\mathcal{L}$ that penalizes the deviations with respect to the observed image I^*, and two regularizers G and R (prior knowledge) on the unknowns (with positive weights α, β). The GBB approach represents images as graphs and employs a skeleton image with only strong gradients as a proxy for I. It uses a re-weighted graph total variation prior $G(I)$ to favor bi-modal image histograms. The PMP method builds on top of the dark-channel prior, proposing a simplified patch-wise minimal pixel prior $G(I)$ that aims for sparse minimal pixel intensities with low computation complexity. The resulting h from each method is Fourier-transformed into the MTF and sampled at the same spatial frequencies as the learning-based approach (Fig. 6), for better comparison. We use the source code from [37], [38], setting the kernel size parameters to 31×31 pixels.

2) Learning-based Method: We upgrade a learning-based approach [39] to *directly* estimate MTF values from natural images (without estimating the kernel h first). It consists of a pre-processing stage followed by a CNN.

The pre-processing stage includes four steps: (i) Intensities are first scaled to $[0, 1]$ and mean-normalized. (ii) A rotation is applied to estimate the MTF in radial and tangential directions. (iii) The Sobel-filtered image patch is passed as an additional channel to aid the MTF estimation procedure. (iv) Channels are spatially down-sampled to enlarge the receptive field of early CNN layers. We alter step (ii) to distinguish between estimations in horizontal and vertical directions to allow a comparison with the GBB and PMP baseline methods.

The CNN consists of a convolutional layer, seven residual blocks with strided convolutions, an intermediate feature representation layer, and three fully connected layers that regress the MTF outputs (bottom branch of Fig. 6). The resulting output consists of eight MTF values in the range $[0, 1] \, \mathrm{lines \, px^{-1}}$ at pre-defined spatial image frequencies.

The training is supervised. In [39], pairs of sharp image patches and PSFs (I, h), synthetic or real, are collected. Their convolution leads to the training samples I^*; the respective MTF samples of the PSFs at the pre-defined frequencies serve as training labels. In contrast to [39], we blurred the sharp images by simulated random defocus and motion blur kernels
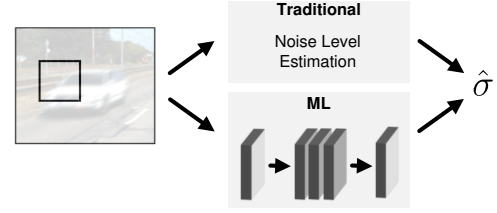

Fig. 7: *Noise estimation* of traditional or learning-based (ML, e.g., multi-layer CNN-based) approaches. Both approaches estimate a noise level $\hat{\sigma}$ for each input image patch.

(see Sec. IV), and retrained the CNN. The original CNN weights are not publicly available and therefore cannot be used for comparison.

At inference time, we pass a batch of four input image patches, i.e., we stack temporally consecutive patches from the same sensor position, pre-process them independently, and input them into the CNN at once. We expect better results this way according to [39], although one patch works as well. The obtained CNN output is then an (averaged) MTF estimation.

Since the original source code is not available, we re-implemented it with guidance from the authors.

E. Noise Estimation

The goal of the image noise estimators is to predict the noise level σ of a noise process given a noisy input image patch $\tilde{I}$, which is monochrome and of size of 128×128 pixels (following the ML approach). Figure 7 depicts the steps of the two main approaches.

1) Traditional methods (non-learning–based): As baseline estimators we use the works of [40] (self-implemented) and [41] (with its code basis [42]). Both are representatives of the two major noise estimation approaches in the literature:

The adaptive Gaussian filtering method [40] (B+F) uses the standard deviation of the most homogeneous image patches as a basis to calculate a Gaussian kernel that is used to filter such patches. The standard deviation of the difference between filtered and unfiltered patches leads to the estimated $\hat{\sigma}$. We increased the internal image patch size from 3×16 to 8×16 pixels as we observed better results on the selected datasets.

The method [41] decomposes image patches via principal component analysis (PCA, also abbreviation of the method) into their eigenvalues and assigns the noise ratio to the smallest ones. In contrast to previous work, the authors tackle the problem of overestimating or underestimating noise theoretically and propose an efficient non-parametric algorithm for noise level estimation.

2) Learning-based Method: We use the work of [43] as learning-based (ML) approach with its code basis [44]. It was designed for pixel-wise noise level estimation from signal-dependent noisy images. The noise model was assumed Gaussian with parameters accounting for photon and readout noise.

The CNN consists of 16 convolutional layers (including three residual blocks) and lacks pooling and interpolation layers, due to a known performance decrease for image noise tasks. The resulting output $\hat{\sigma}$ is estimated for each pixel,

Fig. 8: *Datasets*. Exemplary images from datasets *Sim* (896×768 px), Udacity (1920×1200 px) and KITTI (1242×375 px).

however, for a better comparison with baseline methods, we use the median over the patch as the noise level estimator.

The training in [43] is supervised and carried out by artificially adding noise with $\sigma \in [0, 30]$ DN to images from the Waterloo dataset [45]. We retrained the CNN in the same way using our noise model of Fig. 4 (details in Sec. IV). Applying the original CNN weights to the noise model of [30] and to real images failed; we could only reconstruct the authors' results for their simplified Gaussian noise process.

IV. EXPERIMENTS

We first describe the datasets used and the image corruptions applied (Sec. IV-A). Subsequently, we evaluate the accuracy and runtime performances for the proposed blur and noise estimators separately (Secs. IV-B and IV-C) and on combined image corruptions (Sec. IV-D). All experiments are executed on an Intel Xeon W-2145 CPU and an NVIDIA Quadro RTX 6000 GPU, with the CNN methods running on the GPU.

A. Datasets

We employ one simulated and two real-world datasets: *Sim*, KITTI [46] and Udacity [47] (Fig. 8). We create *Sim* with the simulator [48] to provide accurate ground truth for blur and noise estimation. *Sim* comprises 1000 images of a village environment acquired from different viewpoints and includes vehicles, such as cars and bikes. From KITTI we use the annotated object detection sub-dataset (with preceding frames), and from Udacity we use sub-dataset #2. We subsample KITTI and Udacity for two reasons: to reduce processing time and to remove (in all conscience) clearly visible blur/noise corrupted images that would bias estimation results (however, a residual risk of corruption in the natural images remains). To this end, we pick 1000 images per dataset for noise estimation and 150 images for blur estimation, and match these numbers on *Sim*. For blur, we only use image patches containing detected objects of interest. Note that each dataset yields several image patches, depending on the image size and the blur/noise estimator used (e.g., 1000 Udacity images result in 135k patches for noise estimation).

We chose KITTI and Udacity as the most frequently used ones in the literature for real-world transportation scenarios that fit our requirements, providing a solid comparison baseline, and *Sim* to fully control the proposed blur and noise sources. These datasets can be substituted with others that ideally contain minimal blur and noise, consecutive frames (for more stable median estimations, cf. Secs. IV-C and IV-C), and annotated objects of interest. Modern large-scale datasets (e.g., ROAD [49], Mapillary [50], and Bdd100k [51]) provide a greater variety of scenes and object classes, but typically consist of image sequences from different cameras having diverse geometric and radiometric statistics (e.g., their MTFs can differ significantly). For an overview of more datasets (including KITTI and Udacity), we refer to [52].

All datasets are synthetically corrupted with controlled amounts of noise and blur using the models of Fig. 4.

Noise: Following the "real noise" studies in [26], we generate noise with levels $\sigma \in \{5, 10, 15, 20, 25\}$ DN (digital numbers on a $[0, 255]$ scale). We apply default CMOS camera parameters from [30] and study noise in isolation or in combination. (*i*) For isolated dark current shot noise (DCSN) and readout noise studies, we set the temperature to $T = 330$ K and the exposure time to $t_{exp} = 0.1$ s. (*ii*) For the combined noise case we include *all* noise sources, with random $T \in (300, 330)$ K and $t_{exp} \in (0.002, 1)$ s to emphasize different noise components in each image. In order to reach the desired σ, we amplify the (raw) noise in both settings.

Blur: We synthesize blur kernels of size $d \in \{3, 7, 11, 15, 21\}$ px. d is the diameter for defocus kernels or the approximate path length for motion blur kernels. Defocus blur kernels are calculated analytically (see supplementary material). Motion blur kernels are generated using [53], distinguishing between linear motion kernels (motion intensity parameter set to 0) and non-linear ones (parameter set to 1.0), and manually selecting the kernels that satisfy the target d. We mitigate the influence of motion blur direction by evaluating four rotated versions of each kernel separately (rotating them 0, 45, 90, and 135 deg counterclockwise) and using their average estimation error as final result.

We further propose two use cases for *combined blur and noise* occurrences: (*i*) *Defocus blur and DCSN* (Defocus + DCSN) that might arise at high temperatures (as caused by direct Sun illumination) and with defocus induced by material stress in the optics setup [7], [24], and (*ii*) *photon noise and motion blur* (Photon + Motion) due to high exposure times and signal amplification, typical of low light conditions.

B. Blur Estimation

We assess blur estimation accuracy in terms of the average mean absolute error (AMAE) between a robust MTF estimation ($\widehat{\text{MTF}}$, with $\approx 5\%$ outliers rejected) and ground truth (GT) samples at eight frequencies (f_i) each in horizontal (H) and vertical (V) image directions (w):

$$\text{AMAE} \doteq \frac{1}{2} \sum_{w=\{\text{H,V}\}} \text{MAE}(w),$$

$$\text{MAE}(w) \doteq \frac{1}{8} \sum_{i=1}^{8} \left| \text{MTF}_w^{\text{GT}}(f_i) - \widehat{\text{MTF}}_w(f_i) \right|. \tag{5}$$

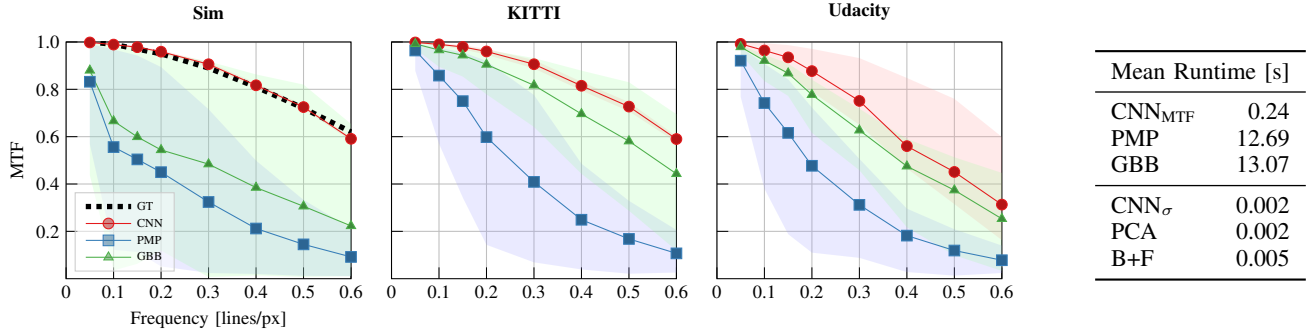| Mean Runtime [s]	
CNN$_{\mathrm{MTF}}$	0.24
PMP	12.69
GBB	13.07
CNN$_\sigma$	0.002
PCA	0.002
B+F	0.005

Fig. 9: *Blur estimation of uncorrupted datasets (i.e., "ground truth")*. Left: Median, minimum and maximum blur estimations of the uncorrupted datasets (depicted by sampled points with interpolation in between and the shaded areas, respectively; horizontal direction only). Right: Mean runtime estimations per image patch (for CNN$_{\mathrm{MTF}}$ per input batch of four images).

TABLE I: *Blur estimation of synthetically corrupted datasets*. Left: Ground truth blur kernels and average mean absolute errors (AMAE) of horizontal and vertical median blur estimations [%]. The best results per kernel and dataset are highlighted in bold. Right: Typical GBB/PMP kernel estimations with undesired artifacts (compare to respective ground truth kernels).

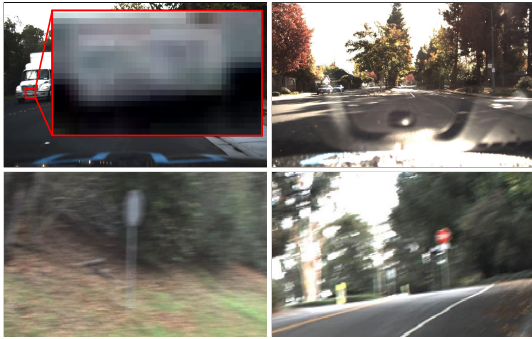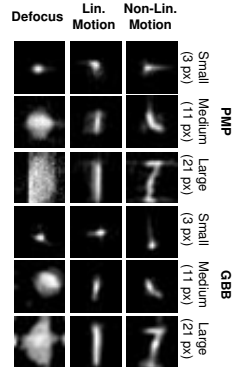|		Defocus Blur					Linear Motion Blur					Non-linear Motion Blur				
Size [px]		3	7	11	15	21	3	7	11	15	21	3	7	11	15	21
Kernel																
Sim	CNN	**0.7**	**1.8**	**2.1**	**0.5**	**1.1**	**6.3**	**10.3**	9.4	9.4	**7.7**	**2.9**	12.2	11.4	19.5	25.0
	PMP	13.9	5.2	3.0	5.3	6.7	37.2	17.8	13.3	**7.3**	16.6	21.8	14.0	13.8	9.8	**11.5**
	GBB	2.7	3.8	6.3	8.4	17.6	31.6	11.4	**7.7**	**7.3**	14.5	17.7	**8.3**	**8.2**	**9.7**	15.0
KITTI	CNN	**0.3**	5.3	2.7	**2.3**	**0.7**	**3.4**	10.9	9.9	9.1	**6.6**	**4.0**	14.1	11.2	14.1	9.2
	PMP	5.8	**2.3**	**1.5**	2.9	4.8	37.2	12.2	8.7	**4.1**	9.5	22.5	7.9	7.3	5.2	4.2
	GBB	3.2	2.9	2.6	**2.3**	9.3	13.4	**5.8**	**5.3**	4.7	7.1	8.5	**4.0**	**5.2**	**3.5**	**3.7**
Udacity	CNN	**2.7**	**0.9**	**0.6**	**0.3**	**1.4**	16.2	10.8	9.8	11.4	**7.9**	9.6	10.3	11.9	16.0	19.2
	PMP	15.1	5.6	4.0	3.9	3.9	34.3	14.2	11.5	**8.1**	12.4	23.5	12.0	**11.0**	8.1	**7.6**
	GBB	2.8	8.8	8.6	8.9	23.2	21.7	12.2	**8.5**	12.2	21.1	10.6	**10.1**	11.6	13.5	16.6

Fig. 10: *Challenging conditions in the Udacity dataset*. From top left: Slight motion blur (3 px) in moving direction, light reflections and two examples of severe motion blur.

We first calculated (robust) median, minimum and maximum estimations for the uncorrupted datasets in Fig. 9. In the *Sim* case, we could determine MTF$^{\mathrm{GT}}$ by evaluating a Siemens Star (generated in the simulator) with the tool [54]. For the real-world datasets however, there are no known GT values, but we expect similar sharp images and hence we plot the estimations for comparison.

Analyzing Fig. 9 we make five major observations: (i) The CNN estimates a nearly ideal MTF with hardly any variance in the *Sim* case and provides similarly confident estimations for KITTI. (ii) Contrary to expectations, the CNN estimates a more uncertain and lower MTF for Udacity. Concerning this, we found challenging effects that influenced the estimation, like frequent windshield reflections and regular slight motion blur in the moving direction, despite our pre-selection of images. The traditional estimators (GBB/PMP) are also affected, producing lower median estimations than for KITTI. (iii) The variances of GBB/PMP shrink from *Sim*, via KITTI towards Udacity. (iv) GBB performs noticeably worse in *Sim*. We ascribe its low median and large variance to the lack of image gradient diversity of the *Sim* dataset (GBB relies on gradients, but strong horizontal edges are scarce in *Sim*). (v) PMP produces generally low estimations and its maxima are far from the GT (*Sim*) or expected GT (real-world) values.

Next, we corrupted the datasets with the generated blur kernels and used the sampled MTFs of the kernels as ground truth. The blur AMAE scores are summarized in Table I. We make the general observation that PMP and GBB —unlike the CNN— usually perform worst for small (3 px) and large (21 px) kernel sizes, respectively. This often manifests in undesired artifacts like smear or cuttings in these estimations (see kernels in Tab. I right). The decreased performance for small blur cases is in agreement with the results from Fig. 9, where GBB and particularly PMP produce lower median estimations and higher variance for *Sim*/KITTI, and lower variance for the already corrupted Udacity. Since GBB/PMP follow a coarse-
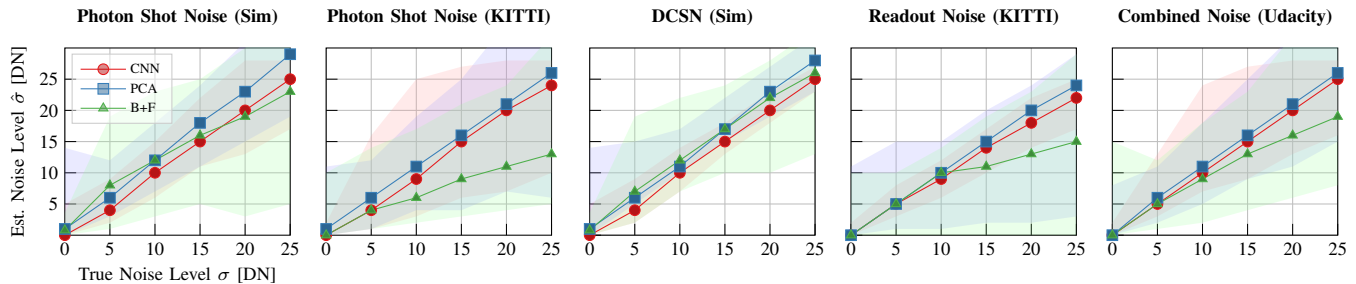
Fig. 11: *Noise estimation of corrupted datasets.* Median, minimum and maximum statistics (depicted by sampled points with interpolation in between and the shaded areas, respectively) of the three proposed noise estimators (CNN, PCA, B+F) as the noise level σ increases (from 0 to 25 grayscale levels, DN), for several types of noise (Photon Shot, DCSN, etc.) and datasets (*Sim*, KITTI, Udacity). The last plot shows the effect of combining all noise types (on the Udacity dataset).

to-fine approach, more internal iterations would enhance the level of detail of the kernel and thus produce smaller errors (at the expense of computational cost). On the other hand, larger kernel estimations improve as larger image patches are used. The authors of GBB [35] suggest kernels be much smaller than the image to have a well-defined blur estimation problem. We further regularly observe larger estimation errors for Udacity. This confirms that Udacity is already corrupted by blur and/or the estimations are influenced by challenging conditions (Fig. 10).

Apart from the already mentioned small/large kernels, all methods estimate defocus well (Tab. I). Nevertheless, the CNN delivers the most accurate results. GBB considers the common simplification of Gaussian blur for defocus, whereas PMP does not and tends to perform slightly better than GBB.

The CNN also estimates linear motion blur comparably well but (except for small/large kernels) GBB tends to produce the smallest errors.

Non-linear motion estimation results (also in Tab. I) differ for the CNN method, which tends to produce larger errors towards the larger (and more complex) kernels compared to the traditional estimators and the linear case. We interpret this as a larger uncertainty and conclude that the CNN might not be appropriate for estimation of complex non-linear motion kernels. In contrast, the scores of GBB/PMP are more accurate among the different kernels and datasets (with GBB a bit better). This slightly better motion blur estimation performance of GBB compared to PMP is consistent with the experiments in [35], where PMP is compared to the work of Pan et al. [55] that first proposes a dark channel prior.

Computational performance: In terms of runtime, we see from the table in Fig. 9 that the CNN executes more than $\times 50$ faster than GBB/PMP and moves in the realm of real-time capability. We also found that CNN requires 98% of its runtime for serial data pre-processing, which can be improved by vectorization. Although the CNN itself executes on a GPU, the running times of current GBB/PMP implementations (running on the CPU) are too long to be practical for a condition monitoring application (especially for multiple image patches).

In summary, the GBB and PMP methods are in general not accurate for blur-free or small/large blur kernel estimation on the image patch sizes used, and available implementations are not real-time capable. Nevertheless, they provide the best

estimates for medium-sized linear and non-linear motion blur kernels. The CNN method, on the other hand, might not be suited for complex non-linear motion kernels, but performs well in terms of defocus, linear motion and real-time requirements. If non-linear motion blur can be circumvented (e.g., with short exposure times or slow motions), the CNN method can be employed for monitoring a camera's condition.

C. Noise Estimation

We evaluate the proposed noise estimators by comparing their robust median, minimum and maximum statistics (rejecting $\approx 5\%$ outliers) against the controlled ground truth noise levels. Results are reported in Fig. 11. Since we obtained comparable results for DCSN and readout noise per dataset, we dropped similar plots.

We first observed that B+F and PCA methods are prone to structural misestimation: both over-estimate low noise levels, and B+F under-estimates high noise levels. These phenomena have been already reported and are characteristic of the corresponding model family [40], [41]. Moreover, all methods tend to strongly under-estimate noise in natural images, which even reduces the median performance of the B+F method. We observe this behavior in over-exposed areas where most pixels are in saturation, which is expected from vehicle camera images containing large sky areas. The CNN method is less vulnerable since it learned employing fewer meaningful pixels; [14] omits such image regions under the assumption that under-/over-saturated patches "cannot contain noise" (which only holds for *completely* saturated regions).

Another observation is the striking difference between the signal-dependent and signal-independent noise cases. signal-dependent photon shot noise increases the variance of all estimators, especially on real-world data. We observed that large variations in bright and dark intensity areas within one image patch led to over- and under-estimation, respectively. The CNN noise level is limited here since it was trained with $\sigma \leq 30\,\mathrm{DN}$. If all noise types occur simultaneously (last plot in Fig. 11) the estimations become more accurate and more robust than in the case of all noise being attributed to photon shot noise. According to the observations of [25], [26], realistic noise follows a combined Poisson-Gaussian distribution, and the Poisson part is troublesome for the noise estimators (in particular for those with Gaussian assumptions). Hence, we
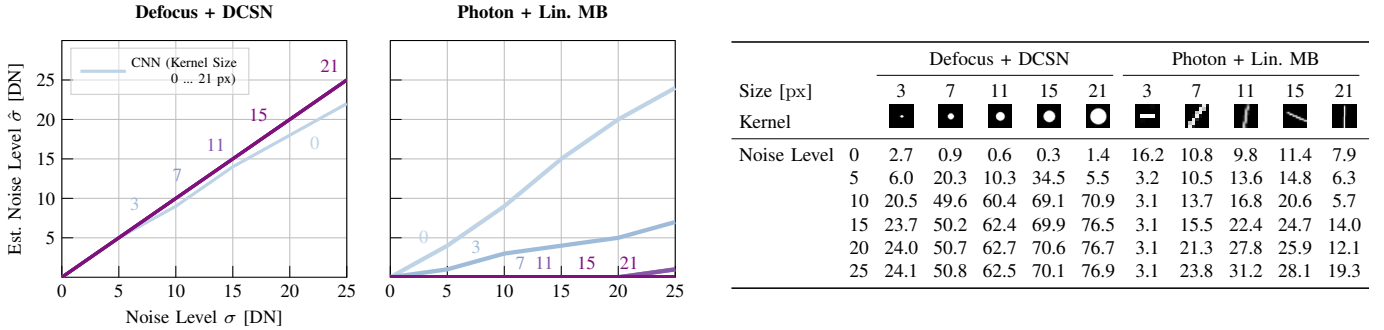
Fig. 12: *Combined estimations of blur and noise* for two image corruption configurations: Defocus + DCSN and Photon + Linear Motion Blur, both on the Udacity dataset. Plots of the median noise estimation (Left) and table with median blur estimation (AMAE (5) in %) (Right) for different noise levels and kernel sizes. Noise estimated for different blur kernel sizes is color coded from blue to purple. However, differences are almost indistinguishable at this scale.

consider isolated photon shot noise as the worst case scenario. The CNN and PCA methods perform similarly if signal-dependent photon shot noise is included, and the CNN is more reliable (smaller variance) otherwise. In terms of denoising, similar results have been shown by comparing traditional and learning-based methods on real data [25].

Computational performance: Regarding runtime (Fig. 9, right table), CNN and PCA executed fastest, with an average of $2\,\mathrm{ms}$ per patch, but in the same order of magnitude as the B+F ($5\,\mathrm{ms}$). All noise estimators are real-time capable and considerably faster than blur estimators.

Summarizing, the CNN and PCA methods are accurate in median but their reliability decreases the stronger the photon shot noise is. In case of signal-independent noise only, the CNN performs by far most reliably. Since PCA is prone to structural misestimation (e.g., over-exposed areas, small noise levels), we suggest using the CNN for condition monitoring applications. Finally, the reliability of PCA and CNN could be improved by using the median estimation from consecutive frames.

D. Combined Estimation of Blur and Noise

Because previous sections showed that CNN blur and noise estimators performed among the best ones on isolated blur/noise cases, we now use these estimators on combined blur and noise corruption experiments. Fig. 12 shows the results for combined defocus blur and DCSN ("Defocus + DCSN"), and Photon Shot Noise with simultaneous linear motion blur ("Photon + Lin. Motion"), both on Udacity.

1) Defocus + DCSN: According to the physics behind the image formation process in Fig. 4, an image is corrupted by defocus first and DCSN afterwards. Hence, high-frequency image content is filtered and fully represented by the DCSN. In theory, the larger the blur the easier the noise estimation. This is what we observe in the first plot of Fig. 12. Although there is a small estimation error for zero defocus, $\hat{\sigma}$ becomes most accurate for $d \geq 3\,\mathrm{px}$ and remains unchanged. Hence, defocus is favorable for DCSN estimation. We expect the same effect for other combinations of defocus/motion blur and DCSN/readout noise.

On the other hand, DCSN negatively affects defocus estimation because advantageous information for detecting blur

(the absence of high frequencies) gets corrupted by noise. We notice two effects from the results on the table of Fig. 12: All defocus estimations worsen with increasing noise levels, and this impact becomes more severe for larger kernel sizes. While estimations for the smallest and largest kernels ($d \in \{3, 21\}\,\mathrm{px}$) can be considered as still good for $\sigma = 10\,\mathrm{DN}$, the same noise level otherwise leads to poor blur estimations. This outcome was investigated in the context of motion deblurring [27], where it was found that, as σ grows, blur estimations approach the Dirac delta function in a large variety of approaches. We observe the same behavior for the CNN estimations, hence the increasing relative error towards larger kernels. Generally, defocus estimations are not robust in presence of subsequent noise. Since sensor noise can be detected accurately in case of defocus, a small $\hat{\sigma}$ should be assured before trusting blur estimations.

2) Photon + Lin. Motion: In this case noise is added before the blur (due to the physics behind the image formation model in Fig. 4). Therefore, we expect the opposite behavior, i.e., a poor noise estimation (the blur kernel acts as a classical noise filter) and a good blur estimation. However, only the noise estimation meets the expectations (see the second plot and table in Fig. 12). A motion blur of size $d = 3\,\mathrm{px}$ already majorly disturbs noise estimation (note that noise is not removed from the image but spread among neighboring pixels). On the other hand, the motion blur leads to structured directional noise (i.e., false image details), which in turn reduces the estimated blur level by increasing $\widehat{\mathrm{MTF}}$ (Fig. 13). This effect intensifies with increasing noise level. Depending on whether blur is overestimated (e.g., for $d = 3\,\mathrm{px}$) or underestimated (e.g., for $d = 11\,\mathrm{px}$) when $\sigma = 0$, the AMAE score decreases or increases for higher noise levels, respectively.

We do not observe the same behavior when we replace motion blur with defocus blur ("Photon + Defocus", Fig. 14), as defocus blur distributes the noise evenly to the neighboring pixels. The noise still influences the blur estimation of the defocus kernel $d = 3\,\mathrm{px}$, however, the effect becomes negligible for larger defocus kernels ($d \geq 7\,\mathrm{px}$).

We build upon this finding and propose a simple approach to suppress high noise in order to re-enable the detection of preceding blur. Specifically, we apply an additional defocus filter to estimate preceding small or medium blur for high
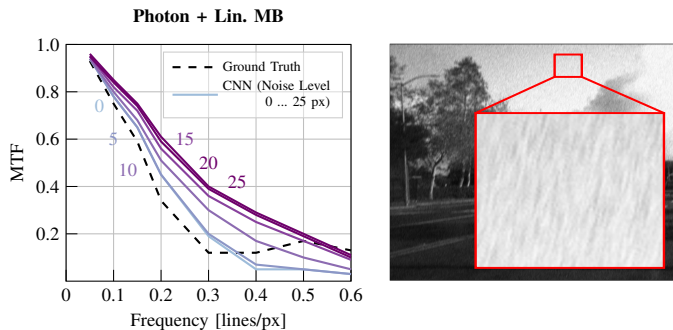
Fig. 13: *Linear motion blur estimation in presence of preceding photon shot noise.* Left: Increasing noise levels σ increase the MTF estimation and thus decrease the estimated blur level d. Right: Corresponding exemplary image with $(d, \sigma) = (11\,\text{px}, 25\,\text{DN})$ showing structured noise induced by subsequent motion blur.

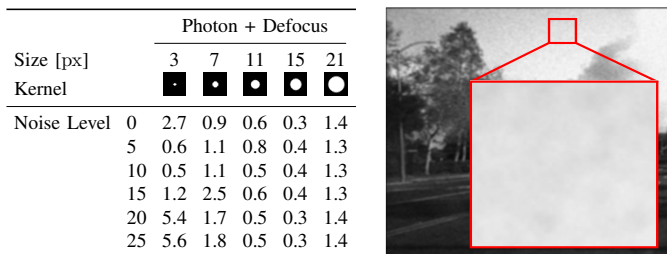|	Photon + Defocus				
Size [px]	3	7	11	15	21
Kernel	▪	◾	◉	◯	◯
Noise Level 0	2.7	0.9	0.6	0.3	1.4
5	0.6	1.1	0.8	0.4	1.3
10	0.5	1.1	0.5	0.4	1.3
15	1.2	2.5	0.6	0.4	1.3
20	5.4	1.7	0.5	0.3	1.4
25	5.6	1.8	0.5	0.3	1.4

Fig. 14: *Defocus blur estimation in presence of preceding photon shot noise.* Left: The minor effect of noise on defocus blur estimation becomes negligible for kernel sizes $d \geq 7\,\text{px}$. Right: Example with $(d, \sigma) = (11\,\text{px}, 25\,\text{DN})$.

sensor noise levels $\sigma \geq 10\,\text{DN}$ (details in the supplementary material). We implemented this improved blur estimation for experiments in Sec. V.

In summary, we conclude that even a small amount of blur boosts the detection of subsequent noise while suppressing preceding noise sources. So, in the presence of blur, photon noise is difficult to estimate and therefore should be avoided. Regarding blur estimation, preceding photon noise can corrupt the result in case of motion blur. Subsequent DCSN with $\sigma \geq 10\,\text{DN}$ already prevents blur estimation, however, it can be re-enabled by applying an additional defocus filter. Hence, if one can eliminate photon noise, we suggest estimating noise before judging a blur estimation result. As in the noise evaluation of Sec. IV-C, sensor noise (DCSN and readout noise) is more favorable than photon shot noise for condition monitoring.

V. MAXIMIZING OBJECT DETECTION BY TRADING OFF BLUR AND NOISE

In this section we demonstrate the application of the proposed framework in a simulated and a real-world scenario (Secs. V-A and V-B, respectively).

Let us first calculate exemplary IOPCs according to Sec. III-C with focus on: (i) object classes *car* and *pedestrian*, (ii) sensor noise only (DCSN + read noise), so that filtered photon noise does not lower the noise level estimation, and (iii) settings from Sec. IV-A for linear motion blur and sensor

noise generation. Figure 15 shows the resulting IOPCs for isolated (left) and combined (right) blur and noise occurrences. It can be seen that the relation from the blur or noise corruptions to the detection performances might be non-trivial and non-linear (e.g., YOLOv4 car detection in presence of Lin. MB) since it is difficult to tell what ML methods learn.

While the proposed IOPC approach is data-driven, it is possible to obtain a model of the object detection AP performance in terms of exposure time (t_{exp}) and ISO gain (A_{ISO}) camera parameters. To this end, we fitted a multivariate function to the IOPCs in Fig. 15, with appropriate basis functions to avoid overfitting while accounting for the IOPSs' non-linearity and non-monotonicity:

$$\widehat{\text{AP}} = c_1 xy + c_2 x + c_3 y + c_4 \sqrt{y} + c_5 + \frac{c_6}{xy} + \frac{c_7}{x} + \frac{c_8}{y}, \quad (6)$$

with the fitted detection score $\widehat{\text{AP}}$, $x \propto A_{\text{ISO}}$, $y \propto t_{\text{exp}}$, and regression coefficients $c_1, \ldots, c_8$. The details of this fitting process are provided in the supplementary material.

A. Example 1

We first demonstrate this framework using the *Sim* environment on a concrete example of YOLOv4 car detection with corrupted data by means of Lin. MB and sensor noise (Fig. 16). The left image in Fig. 16 depicts the scene in uncorrupted conditions (without noise or blur), for reference. Here the first car is detected fairly ($p = 0.53$) and the second one much better ($p = 0.97$). While the CNN noise estimator detects a small noise level of $\hat{\sigma} = 1\,\text{DN}$ by mistake, the MTF estimation is nearly ideal ($\widehat{\text{MTF}} = 0.99$). Next, we included a realistic trajectory for the simulated camera to create a linear motion with a speed of $v \approx 760\,\text{px/s}$. This causes blur (an exposure time of $t_{\text{exp}} = 4\,\text{ms}$ induces a motion blur of $d_{\text{old}} \approx 3\,\text{px}$), and we also apply sensor noise of $\sigma = 20\,\text{DN}$. In this situation (second image in Fig. 16) blur and noise are estimated within the expected error ranges ($\hat{d}_{\text{old}} \approx 3\,\text{px}$), but the cars are detected worse ($p \approx 0.25$).

In the next step, we determine $\alpha^\star$: knowing the relation between motion blur sizes and estimated MTFs (first plot in Fig. 15) and the estimated noise level, we target an $\widehat{\text{MTF}} \in [0.7, 0.8]$, which corresponds to approximately $\hat{d} \in [11, 12]\,\text{px}$ (cf. first heat plot in Fig. 15). We chose $d_{\text{target}} \approx 12\,\text{px}$, hence, $\alpha^\star = \hat{d}_{\text{old}}/d_{\text{target}} \approx 3/12 = 0.25$. We then reduce the ISO gain by the factor $\alpha^\star$ and show an intermediate image without increasing t_{exp}. One car is now detected more confidently while blur and noise are still estimated within the expected error ranges. As we did not investigate the influence of image intensity on object detection performance, next we increase t_{exp} by the factor $\alpha^\star$ to restore the original intensity level, producing the last image of Fig. 16. In this last step the total detection score slightly increases despite the likewise motion blur amplification ($d \approx \hat{d} = 12\,\text{px}$). The steps taken are marked with red arrows on the heat plot in Fig. 15.

B. Example 2

For a real-world example, we deployed our framework on a real camera system using an Allied Vision Prosilica GC1380H

Linear Motion Blur | **Combined Noise** | **Lin. MB + Sensor Noise: YOLOv4 Car** | **Lin. MB + Sensor Noise: Faster R-CNN Pedestrian**
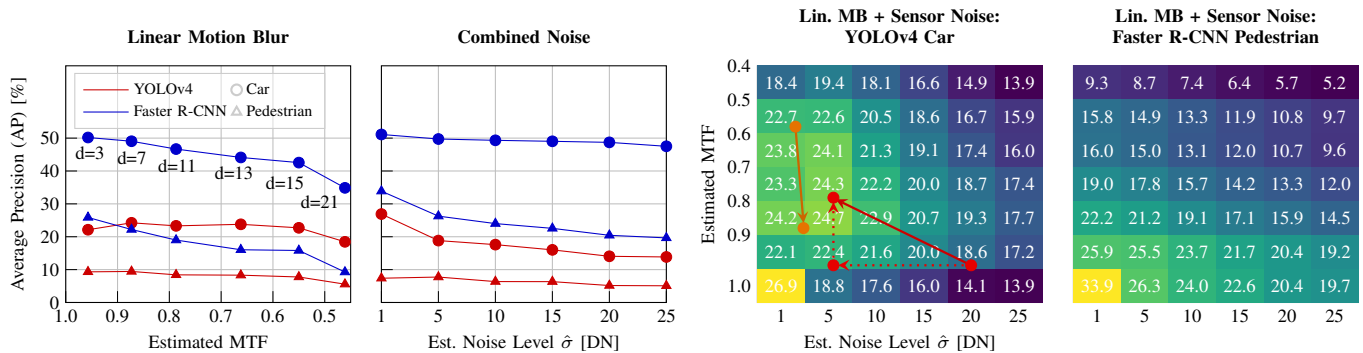
Fig. 15: *Influence of blur and noise on object detection performance.* Exemplary object detection performances depending on isolated (Left) and combined (Right) occurring and blur and noise. All input-output profiles depend on the actual estimated corruption levels. Performances are measured in terms of average precision (AP). Noise levels and MTFs are estimated by the respective CNN methods and the MTFs depict means for horizontal and vertical measurements at frequency $f = 0.1$. The red and orange arrows demonstrate two examples of exposure time t_{exp} / ISO-gain trade-off paths (see text in Sec. V).

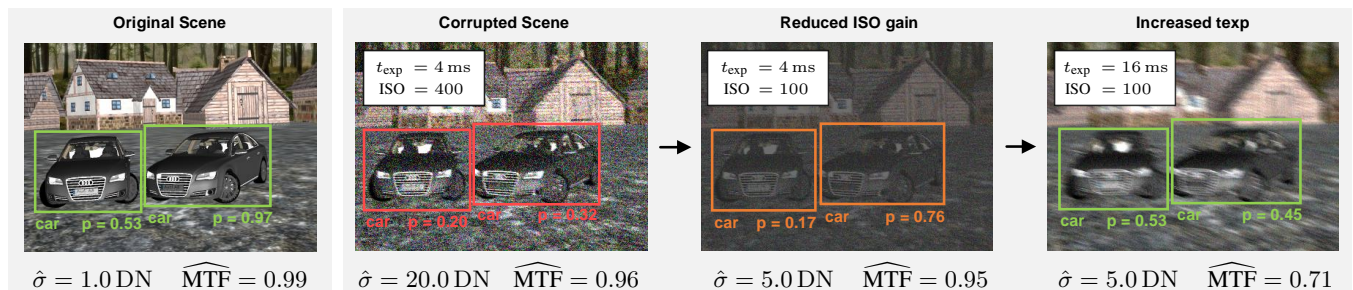| Original Scene	Corrupted Scene	Reduced ISO gain	Increased texp
$\hat{\sigma} = 1.0\,\text{DN}$ $\widehat{\text{MTF}} = 0.99$	$\hat{\sigma} = 20.0\,\text{DN}$ $\widehat{\text{MTF}} = 0.96$	$\hat{\sigma} = 5.0\,\text{DN}$ $\widehat{\text{MTF}} = 0.95$	$\hat{\sigma} = 5.0\,\text{DN}$ $\widehat{\text{MTF}} = 0.71$

Fig. 16: *Maximizing object detection by trading off blur and noise.* Application of the proposed framework to detect cars using YOLOv4 on *Sim* data suffering from linear motion blur and sensor noise. The scene (Left) is first imaged with an ISO gain of 400 (leading to sensor noise of $\sigma \approx 20\,\text{DN}$) and an exposure time of $4\,\text{ms}$. As a result, the car recognition performance of YOLOv4 decreases drastically (Center-Left). Applying the optimal $\alpha^\star \approx 0.25$ (according to the performance profile from Fig. 15) improves car detection (Center-Right). Finally, we divide the exposure time by $\alpha^\star$ to compensate for the missing light, which improves overall detection slightly (Right). Hence, noise is reduced from $\sigma \approx 20\,\text{DN}$ to $5\,\text{DN}$ and blur is increased from $d \approx 3\,\text{px}$ to $12\,\text{px}$ while detection rate increases from $p \approx 0.25$ to $p \approx 0.5$.

Camera System (Prosilica GC1380H)

Mobile Computer (CPU: Intel i7-9850H, GPU: NVIDIA MX150)

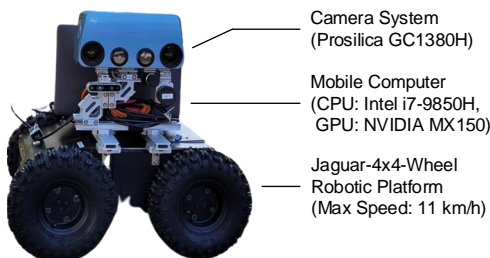Jaguar-4x4-Wheel Robotic Platform (Max Speed: 11 km/h)

Fig. 17: *Deploying our framework on a real camera system* that is attached on an autonomous mobile robotic platform.

camera [56] attached on a Jaguar-4x4-wheel mobile robotic platform [57] and a mobile computer doing the real-time calculations (Fig. 17). With this setup, we demonstrate another example of the non-monotonic YOLOv4–car-detection heat map of Fig. 15, marked with an orange arrow. We therefore navigated the camera system through a low-illuminated parking lot with fixed initial camera parameters $t_{exp} = 8\,\text{ms}$, ISO = 100, and default values for the rest. To make the YOLOv4–car-detection profile applicable, we target (i) a constant linear

motion blur induced by a constant speed of the platform and (ii) a low noise level with the low ISO gain to neglect the undesired impact of photon shot noise. Subsequently, the experiment was repeated with the camera's built-in t_{exp} / ISO gain controller [56] and compared to ours with respect to response time and object detection performance (Fig. 18). Both camera parameter controllers were automatically triggered on the first image frame at $t = 0\,\text{ms}$. Finally, we sampled the video sequences for each configuration to contain 200 ± 5 cars, which we manually annotated for YOLOv4.

The built-in camera controller tracks a mean image intensity level of 50% and prioritizes changing t_{exp} over ISO gain as long as $t_{exp} \leq 500\,\text{ms}$. Hence, the built-in controller constantly changed t_{exp} only, did not account for the motion blur, and resulted in an AP car detection score of 26.08%.

Our proposed framework took about $3000\,\text{ms}$ to estimate $(\hat{\sigma}, \widehat{\text{MTF}}) = (0.1, 0.57)$ (longer than in Sec. IV due to the weaker mobile hardware, but still interactive / real-time). With initially fixed camera parameters (i.e., while $t < 3000\,\text{ms}$), YOLOv4 reached an AP score of 47.54%. The system then decided to decrease the motion blur at the expense of slightly

$\hat{\sigma} = 0.3\,\mathrm{DN}$ $\widehat{\mathrm{MTF}} = 0.71$ $\hat{\sigma} = 0.3\,\mathrm{DN}$ $\widehat{\mathrm{MTF}} = 0.61$

$\hat{\sigma} = 0.2\,\mathrm{DN}$ $\widehat{\mathrm{MTF}} = 0.57$ $\hat{\sigma} = 0.4\,\mathrm{DN}$ $\widehat{\mathrm{MTF}} = 0.90$
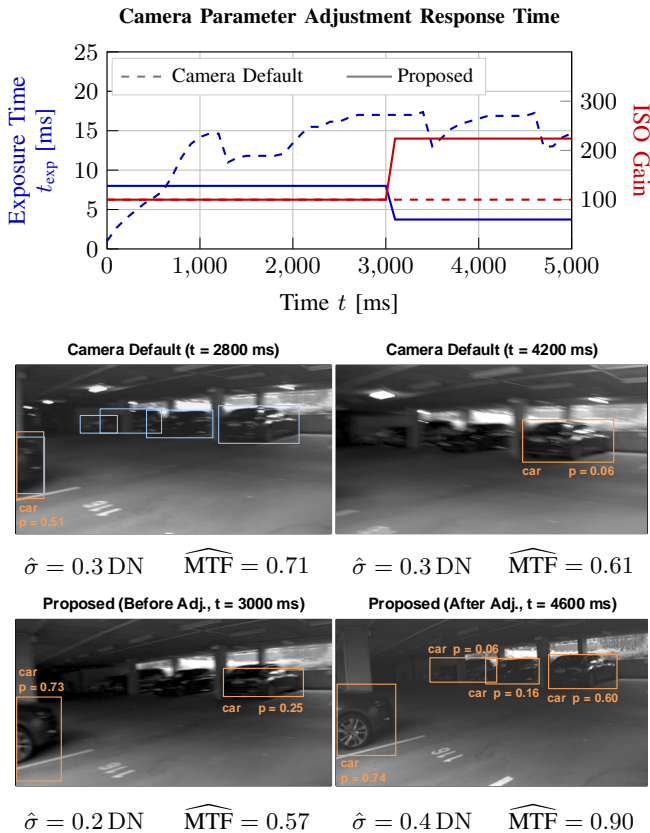
Fig. 18: *Comparison of built-in camera t_{exp} / gain control vs. our framework.* Diagram: The built-in controller constantly optimized image intensity by adjusting t_{exp} only. In contrast, our framework targets optimal object detection performance and adjusted t_{exp} / gain once. Images: Examples from the experiments (adapted brightness and contrast for better visualization). Blue boxes indicate ground truth objects, and orange boxes actual detections. The overall AP scores were: 26.08% for the built-in camera control, 47.54% for the manually chosen fixed parameters (before automatic adjustment), and 60.56% for our framework (after automatic adjustment).

increasing the noise to move to higher AP detection values (brighter part of the heat map). Inspecting the AP curves (1st plot in Fig. 15), $\mathrm{MTF} = 0.57$ corresponds to $d = 15\,\mathrm{px}$, and the system targeted $\widehat{\mathrm{MTF}} \in [0.8, 0.9]$ (high values of the heat map), which corresponds to a smaller motion blur of $\hat{d} \approx 7\,\mathrm{px}$. Two steps were taken: first, the system decreased the exposure time by a factor $\alpha = 15/7 \approx 2.14$ to achieve the desired MTF improvement. Intel Then, it increased the ISO (and increased noise) by the same factor $\alpha \approx 2.14$ to restore the intensity level for the detector. The final operating point was $(\hat{\sigma}, \widehat{\mathrm{MTF}}) \approx (0.4, 0.9)$, which has a higher AP value (60.56%) than the initial point.

VI. CONCLUSION

We have proposed a framework for real-time camera conditioning, bringing together the tasks of inferring the state of the system and acting on the camera's operating point to achieve optimal application performance. Our framework has a modular design, hence it is flexible and interpretable, allowing for multiple choices of its submodules, such as the image quality estimators. To this end, we have carried out a comprehensive experimental study close to the physics of the sensor and have incorporated six state-of-the-art image quality estimators, two advanced object detectors and two standard datasets plus one self-created. We have considered a more extensive and realistic image formation pipeline than preceding works by including motion and defocus blur as well as simultaneous occurring corruptions that influence each other. All these elements have been put together in a coherent manner to justify our design choices and provide insights and practical recommendations with regard to camera monitoring applications (summarized at the end of each experimental subsection).

Regarding the framework, the main idea is that aiming at improving image quality blindly, without taking into account the subsequent high-level application, may not always be the best. If the end goal is better high-level application performance (say, car detection), then it is sensible to trade off image quality for whole system performance by adjusting the camera parameters. We have demonstrated this on how image blur and noise (image quality) affect an object detection application and have implemented it on a real-world ground robot; the specific control strategy of the camera parameters (exposure time and ISO gain) depends on the experimental input-output performance curves of the object detector (which is in general non-linear and non-monotonic). However, our framework is generic: it is not limited to the proposed control strategy (one could control a motor to adjust focus), it can be applied to other optical sensor systems (as infrared or event cameras), other scenarios, and it can consider other "features", conceivably application-specific, besides blur and noise. These have been selected because they are among the most generic and influential effects in image processing. The framework can also be easily extended to optimize multiple vision applications running simultaneously (e.g., by maximizing their weighted average performances).

Lastly, we have focused on a subset of corruptions originating in the camera itself. A possible extension would be to model additional camera conditions and retrain the estimators accordingly (e.g., more sophisticated corruptions such as defocus due to heat-induced material stress or conditions originating outside the camera, like low/high scene illumination), and compensate for them by triggering actuators (e.g., cooling or headlights). This could also require the acquisition and exploitation of additional data, such as the camera's and environment's configuration (focal length, aperture size, exposure time, temperature, positioning, illumination), leading to the research and development of more advanced Sensor AI approaches [58].

The following supplementary material complements the main paper. We first detail the assumed image formation process (Sec. VII) and subsequently propose a simple approach to improve blur estimations in the presence of high noise (Sec. VIII). Finally, we provide two additional experiments for the sake of completeness: a performance curve analysis of the latest YOLOv7 object detector (Sec. IX) and a data-driven model derivation for YOLOv4 object detection performance as a function of exposure time and ISO gain camera parameters (Sec. X).

VII. IMAGE FORMATION PROCESS

The image formation process that we consider in a standard camera is depicted in Fig. 4. Let us specify the image blur and noise components of this model, and metrics to quantify them.

A. Blur

Image blur is the result of processes that reduce image sharpness. The most prominent of such processes are (i) light refracted by a defocused lens, (ii) motion between the sensor and the scene, (iii) atmospheric turbulence, and (iv) diffraction [59, p. 325]. We focus on the former two sources, whose induced blur types are known as defocus and motion blur, respectively. Many factors contribute to these processes and make their mathematical description complex. For the sake of simplicity they are often modelled as a convolution on the image plane:

$$I^*(x,y) = I(x,y) \circledast h(x,y), \tag{7}$$

where $I(x,y)$ is the input intensity at pixel (x,y) (before the blur process), $h(x,y)$ is the blur kernel and $I^*(x,y)$ is the blurred image intensity. The kernel $h(x,y)$ is also called point spread function (PSF) [59, p. 328].

The PSF can be used to objectively quantify image blur. Its Fourier transform is the optical transfer function (OTF) and it describes how spatial frequencies f (i.e., image details, contrast) are affected by blur:

$$\mathrm{PSF}(x,y) \quad \overset{\mathcal{F}}{\mapsto} \quad \mathrm{OTF}(f) \propto \mathrm{MTF}(f)\, e^{i\,\mathrm{PhTF}(f)}. \tag{8}$$

Usually only the magnitude of the OTF, known as the modulation transfer function (MTF), is relevant to quantify blur, and so the phase transfer function (PhTF) is omitted. Let us now describe defocus and motion blur kernels $h(x,y)$.

1) Defocus Blur: We assume a defocus blur kernel $h(x,y)$ that distributes a pixel's intensity evenly over an approximate circular area of neighboring pixels (with radius r and center (c_x, c_y)) [59, p. 325]:

$$h(x,y) = \begin{cases} s, & (x - c_x)^2 + (y - c_y)^2 \leq r^2 \\ 0, & \text{otherwise,} \end{cases} \tag{9}$$

with the value s determined by the normalization constraint $\iint h(x,y)\,dx\,dy = 1$. This circle refers to the term circle of confusion, whose diameter $d = 2r + 1$ can be calculated as

$$d = A\, \frac{\mathrm{f}}{S_1 - \mathrm{f}}\, \frac{|S_2 - S_1|}{S_2}, \tag{10}$$

expressed in terms of the focused object distance (S_1), the out-of-focus object distance (S_2), the focal length (f), the image distance (f_1) and the aperture diameter (A) [60, p. 216]. This defocus blur kernel model has shown to be on par with more complex models in image reconstruction [61]. We assume the camera comprises a single, perfect, convex, thin lens satisfying $1/\mathrm{f} = 1/\mathrm{f}_1 + 1/S_1$.

2) Motion Blur: Depending on the type of motion, image blur can manifest as translation, rotation, scale changes or a combination of all of them. Hence, a closed-form expression for $h(x,y)$ may be complex to obtain. Its main influencing factors are the exposure time and the relative angular speed between the imaged objects and the sensor during the exposure (see [59, p. 326] for an exemplary approximation of h in a simple scenario). We model $h(x,y)$ to contain a coherent path of pixels with non-zero and potentially inhomogeneous intensities. We assume the path in $h(x,y)$ may be non-linear, since factors like an uneven driving ground and unpredictable moving scene objects might lead to complex non-linear movements during the exposure interval. For simplicity we neglect additional influences like the camera's readout procedure, the influence of the shutter or rapidly changing lightning conditions.

B. Noise

Image noise denotes "any undesired information that contaminates an image" and often occurs during image acquisition or transmission [59, p. 348]. Having the online condition monitoring approach in mind, we tackle the problem of online characterization and mitigation of image acquisition noise. We consider time-varying sources because time-invariant noise sources (such as photo response non-uniformity) are often addressed during calibration (before acquisition) and their residuals are assumed to have a minor influence on image quality. Generally, noise can be modelled by:

$$\tilde{I}(x,y) = I(x,y) + I(x,y)^\gamma\, u(x,y), \tag{11}$$

where $I(x,y)$ is the clean intensity (the signal's intensity), $u(x,y)$ is a random, stationary and uncorrelated noise process, and $\tilde{I}(x,y)$ is the corrupted intensity. A parameter γ controls different noise types. The amount of noise (or noise level) may be quantified using the standard deviation σ of the underlying statistical distribution of $u(x,y)$.

Let us now detail the most prominent time-varying noise processes, namely photon shot noise, dark shot noise and readout noise (Fig. 4). As a theoretical guide, we follow [30].

1) Photon Shot Noise: As photons arrive at the sensor, the counting process within the exposure interval undergoes random fluctuations. This is known as shot noise and follows a Poisson distribution. If the number of arriving photons k is large enough (i.e., in non-low illumination conditions), the Poisson distribution may be approximated by a Gaussian distribution using the Central Limit Theorem [62, p. 225]:

$$\mathcal{P}_\lambda(k) = \frac{\lambda^k}{k!} e^{-\lambda} \quad \overset{k \to \infty}{\approx} \quad \frac{1}{\sqrt{2\pi\lambda}} e^{-(k-\lambda)^2/2\lambda}, \tag{12}$$

with $\lambda = \sigma^2$ as the expected value and variance of the arrival events. The higher the number of arriving photons, the higher

the number of random fluctuations; hence photon shot noise behaves signal-dependent and can be described by (11) when setting $\gamma = 1$ and $u(x, y) \sim P_\lambda(k)$.

2) Dark Current Shot Noise (DCSN): Similar to photon shot noise, dark current (DC) shot noise originates from the random arrival of DC electrons and follows the same distribution (12). DC emerges from thermally generated electrons at different sensor material regions. The amount of generated electrons depends, among others, mainly on the pixel area, temperature and exposure time [63, ch. 7.1.1]. DCSN is signal-independent, hence $\gamma = 0$ in (11).

3) Readout Noise: Readout noise refers to the imperfections due to the sensor's electronic circuitry converting charge into digital values and it is attributed to the on-chip amplification and conversion processing units [64, p. 197]. Although readout noise can be reduced to a negligible level in scientific cameras, its impact is still significant for industry-grade sensors that lack of noise reduction [63, ch. 7.2.9]. We incorporate sense node reset noise (alias kTC noise) and source-follower noise as the main time-varying components.

Both noise sources can be modelled as a zero-mean Gaussian process, where σ mainly depends on the temperature. The overall readout noise contribution is a signal-independent addition of both noise processes, hence $\gamma = 0$ in (11). We keep it at this level of abstraction and refer to [30] for details.

In summary, we consider the blur and noise sources in Fig. 4 and have described their physical models mathematically.

VIII. IMPROVED BLUR ESTIMATION IN PRESENCE OF HIGH NOISE

The Sec. IV-D has pointed out that blur is not accurately estimated in the case of high subsequent noise (e.g., DCSN, with $\sigma \geq 10\,\mathrm{DN}$). Here we demonstrate a simple approach to improve the accuracy of such MTF estimates (Fig. 19). The approach exploits that preceding photon noise is not expected to significantly influence the MTF estimation of subsequent defocus blur (see Sec. IV-D). Hence, the approach consists of considering the above-mentioned "high subsequent noise" as the preceding noise of a new blur stage, estimating the overall MTF and reassigning the credit between the two blur stages. Specifically, following up on the Defocus + DCSN case in Sec. IV-D, the considered pipeline has now three stages: Lin. MB + DCSN + defocus filtering. Letting the first blur kernel be b_1, we filter noise by an additional kernel b_2, estimate the overall blur $\widehat{\mathrm{MTF}}(b_1, b_2) = \widehat{\mathrm{MTF}}(b_1)\,\widehat{\mathrm{MTF}}(b_2)$ and lastly divide the MTF by the known $\mathrm{MTF}^{\mathrm{GT}}(b_2)$ according to the Fourier convolution theorem [65, p. 242]. To this end, we assume $\mathrm{MTF}^{\mathrm{GT}}(b_2) \approx \widehat{\mathrm{MTF}}(b_2)$ and determine the estimation error of $\widehat{\mathrm{MTF}}(b_1)$ with respect to $\mathrm{MTF}^{\mathrm{GT}}(b_1)$.

Due to the combinatorial complexity of the experimental configuration, we focus on the following one grounded in the results from Sec. IV-D: We employ only the CNN method for MTF estimation on Udacity data, and in preparation for Sec. V, we consider the case of Lin. MB + DCSN (representative for sensor noise, to keep it clear and concise). The operating points for this experiment rely on three reasons: (i) The choice of b_2's size (d_2) is a trade-off between filtering
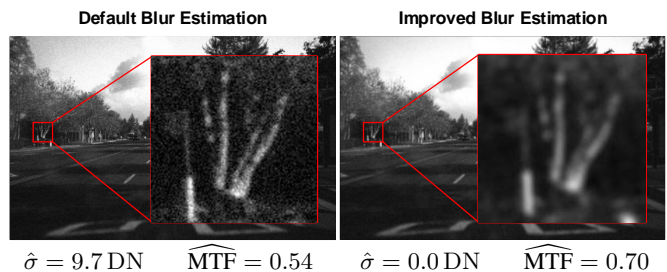

Default Blur Estimation Improved Blur Estimation

$\hat{\sigma} = 9.7\,\mathrm{DN}$ $\widehat{\mathrm{MTF}} = 0.54$ $\hat{\sigma} = 0.0\,\mathrm{DN}$ $\widehat{\mathrm{MTF}} = 0.70$

Fig. 19: *Proposed improved blur estimation in presence of high noise,* in an exemplary scenario of Lin. MB + DCSN with $d_1 = 11\,\mathrm{px}$ and $\sigma = 10\,\mathrm{DN}$. We target estimating the lin. MB with a ground truth of $\mathrm{MTF}^{\mathrm{GT}} = 0.75$ (combined image directions at $f = 0.1$). Left: Noise distracts the blur estimation ($\widehat{\mathrm{MTF}} = 0.54$). Right: Defocus filtering the noise with $d_2 = 7\,\mathrm{px}$ assists the blur estimation ($\widehat{\mathrm{MTF}} = 0.70$, the influence of defocus was cancelled out during the estimation).

TABLE II: *Estimation of linear motion blur* b_1 *(Lin. MB) on combined pipeline (Lin. MB + DCSN + Defocus), using Udacity data. The table reports mean absolute errors (MAE) of horizontal (H) and vertical (V) estimations, their average (AMAE), and their expected values (AMAE$^{\mathrm{Exp.}}$ (13)).*

Corruption Levels		Error Metrics			
d_1 [px]	σ [DN]	MAE (H)	MAE (V)	AMAE	AMAE$^{\mathrm{Exp.}}$
3	10	2.9	2.0	2.5	16.2
3	25	3.5	2.1	2.8	16.2
7	10	12.3	7.7	10.0	10.8
7	25	11.6	8.8	10.2	10.8
11	10	11.1	8.4	9.7	9.8
11	25	14.1	10.1	12.1	9.8
3	10	1.5	0.3	0.9	16.2
3	25	1.7	0.3	1.0	16.2
7	10	14.7	12.7	13.7	10.8
7	25	14.8	12.8	13.8	10.8
11	10	18.6	14.1	16.4	9.8
11	25	20.2	15.1	17.6	9.8

(Left row-group label: $d_2 = 7\,\mathrm{px}$; lower row-group label: $d_2 = 11\,\mathrm{px}$)

the noise to reduce its influence on blur estimation without loosing image details necessary to determine b_1. Hence, we pick the smallest defocus filters $d_2 \in \{7, 11\}\,\mathrm{px}$ that lead to stable blur estimation (cf. Fig. 14). (ii) We consider small/medium motion blur $d_1 \in \{3, 7, 11\}\,\mathrm{px}$ so that the overall blur is still detectable by the CNN. (iii) We focus on severe high/higher noise levels $\sigma \in \{10, 25\}\,\mathrm{DN}$. We next evaluate $\widehat{\mathrm{MTF}}(b_1) \approx \widehat{\mathrm{MTF}}(b_1, b_2)\,/\,\mathrm{MTF}^{\mathrm{GT}}(b_2)$ with Table II.

We need to ensure three preconditions to divide $\mathrm{MTF}^{\mathrm{GT}}(b_2)$ from $\widehat{\mathrm{MTF}}(b_1, b_2)$ for a meaningful result: (i) $\widehat{\mathrm{MTF}}(b_1, b_2) \leq \mathrm{MTF}^{\mathrm{GT}}(b_2)$, (ii) $\mathrm{MTF}^{\mathrm{GT}}(b_2) > 0 + \epsilon$ and (iii) $\widehat{\mathrm{MTF}}(b_1, b_2) > 0 + \epsilon$, for all sampled frequencies. We chose the control parameter $\epsilon = 0.05$ to avoid large quotients for small values, and omit frequencies that do not satisfy the conditions.

Table II presents results in terms of MAE and AMAE scores (5), and their expected values

$$\mathrm{AMAE}^{\mathrm{Exp.}} \doteq \sqrt{\mathrm{AMAE}(\widehat{\mathrm{MTF}}(b_1))^2 + \mathrm{AMAE}(\widehat{\mathrm{MTF}}(b_2))^2} \quad (13)$$

from the error propagation of $\widehat{\mathrm{MTF}}(b_1)$ and $\widehat{\mathrm{MTF}}(b_2)$.

We observe generally worse MAE scores in horizontal than in vertical image direction, which are in agreement with the already-mentioned slight motion blur in the moving direction on Udacity data (the moving direction is closest to the horizontal image axis; see Sec. IV-B). It can be also seen that the higher the considered noise and blur levels, the worse the estimations of b_1. The impact of higher noise, which relativizes with increasing d_1, is in agreement with the results of Fig. 14. Higher blur levels d_1 or d_2 increase the loss of information (where the MTF drops below zero) and thus worsen estimations of b_1. This is also why the smaller defocus $d_2 = 7\,\mathrm{px}$ performs better (with results closer to their expected values) and smaller motion blurs d_1 are estimated more accurately (despite their higher expected values). Moreover, the information loss causes the CNN to generally overestimate d_1, which in turn limits the estimation error for $d_1 = 3\,\mathrm{px}$ as its MTF values for the considered frequencies are already close to one. All in all, a defocus filter with $d_2 = 7\,\mathrm{px}$ has been shown to be the best working solution to restore a blur estimation of d_1 in presence of high noise.

Summarizing, additional defocus filtering suppresses noise so that estimation of preceding small or medium blur can be re-enabled for high sensor noise levels $\sigma \geq 10\,\mathrm{DN}$. This procedure is also suitable for a condition monitoring application as it can be applied in the background without changing the camera configuration.

IX. Blur direction dependence of YOLOv7

The YOLOv4 car performance curves show non-linearities and non-monotonicity in terms of blur and noise (Fig. 15). We examined the latest YOLOv7 object detector [32] for the same effect and found unexpected behavior when blur direction is incorporated as a third dimension.

From Tab. III, it can be observed that car detection performance tends to increase for blur in horizontal image direction (compare values within a column for a fixed noise level). The larger the kernel size, the more the performances vary across the different blur directions. This influence of blur direction can even increase the detection performance as the blur size increases (see red values per row in Tab. III).

We attribute this observation to the MS Coco dataset [66] used to train YOLOv7 [32]. Since natural motion blur occurs primarily for objects moving in horizontal image direction (where the angle between the camera and a scene object is most favourable for motion blur, cf. [59, p. 326]), a dataset of natural images is likely to contain more examples of horizontal motion blur. Thus, the training dataset may be biased, which would also affect YOLOv7. Future studies could balance datasets for the directions in which motion blur occurs.

X. Influence of Camera Parameters on Object Detection Performance

Object detection AP performances can be expressed as a function of exposure time (t_{exp}) and ISO gain (A_{ISO}) camera parameters, if static camera and environmental conditions can be assumed (i.e., constant blur and noise statistics). To this end, we apply a polynomial least-squares regression to the IOPCs

TABLE III: *YOLOv7 car detection performance as a function of blur direction, blur size, and noise level (σ).* Rotation angles $\phi \in \{0, 45, 90, 135\}\,\mathrm{deg}$ are applied counter-clockwise. Red colours indicate row-wise outliers that violate monotonicity.

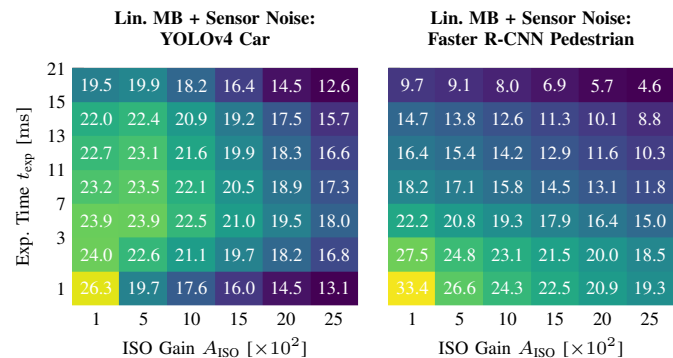|		Linear Motion Blur				
Size [px]		3	7	11	15	21
Kernel		▬	◨	▮	◲	▯
$\sigma = 0\,\mathrm{DN}$	0	50.31	49.68	43.52	44.38	27.17
	45	50.83	49.97	48.19	48.03	38.81
	90	50.80	50.56	49.02	41.57	43.57
	135	50.89	50.70	50.71	41.60	37.49
$\sigma = 5\,\mathrm{DN}$	0	50.60	50.05	43.12	45.91	25.80
	45	49.93	46.58	46.59	45.65	38.33
	90	50.16	49.39	47.58	39.38	41.76
	135	49.60	48.48	48.42	39.00	37.76
$\sigma = 10\,\mathrm{DN}$	0	49.64	48.75	41.76	44.63	24.10
	45	48.31	44.16	44.80	43.35	36.30
	90	47.59	47.37	45.80	37.64	40.29
	135	48.13	46.45	45.95	37.84	36.26
$\sigma = 15\,\mathrm{DN}$	0	48.46	47.07	40.38	43.29	23.31
	45	45.71	42.36	42.55	41.13	33.77
	90	46.07	45.97	43.72	35.39	38.26
	135	45.37	45.14	42.94	35.52	32.65
$\sigma = 20\,\mathrm{DN}$	0	47.19	46.57	39.90	42.57	21.55
	45	44.85	40.97	39.58	39.10	31.37
	90	44.09	42.79	42.29	33.18	36.09
	135	44.13	43.18	41.29	33.56	30.53
$\sigma = 25\,\mathrm{DN}$	0	45.14	44.23	38.98	41.66	21.89
	45	42.47	37.68	37.34	36.78	27.47
	90	42.64	40.37	39.90	30.98	34.80
	135	42.43	41.35	38.58	30.59	28.49

Fig. 20: *Influence of exposure time (t_{exp}) and ISO gain (A_{ISO}) on object detection performance (AP).* Regressed from the curves in Fig. 15 using (6) with coefficients from Tab. IV.

from Fig. 15 with the constraint to keep the degree of the polynomial low (to avoid overfitting) while accounting for the IOPSs' non-linearity and non-monotonicity. We identify the different components of the polynomial manually by mapping the slopes of the IOPCs into respective terms. This yields (6):

$$\widehat{\mathrm{AP}} = c_1 xy + c_2 x + c_3 y + c_4 \sqrt{y} + c_5 + \frac{c_6}{xy} + \frac{c_7}{x} + \frac{c_8}{y} \quad (14)$$

with $x = \lambda_1 A_{ISO}$, $y = \lambda_2 t_{exp}$, and regression coefficients $c_1, \ldots, c_8$ (values in Tab. IV). Note that λ_1 and λ_2 must be calibrated beforehand to match occurring blur and noise statistics to the camera parameters. We calibrated both so

TABLE IV: *Coefficients to fit* (6) *to the curves in Fig.* 15. χ^2 ($\downarrow$) denotes the goodness-of-fit and R^2 ($\uparrow$) the coefficient of determination score.

Param.	YOLOv4 Car		Faster R-CNN Pedestrian	
	Value	Std. Dev.	Value	Std. Dev.
c_1	-5.5862×10^{-5}	2.4493×10^{-5}	4.0763×10^{-5}	2.4130×10^{-5}
c_2	-2.6957×10^{-3}	3.1564×10^{-4}	-3.1039×10^{-3}	3.1096×10^{-4}
c_3	-1.1775	1.915×10^{-1}	-4.9352×10^{-1}	1.8868×10^{-1}
c_4	6.6093	1.3270	-2.7007	1.3074
c_5	1.6871×10^1	2.4239	3.3140×10^1	2.3879
c_6	9.8229×10^2	1.2080×10^2	7.7306×10^2	1.1901×10^2
c_7	-2.9259×10^2	5.6069×10^1	-6.4604×10^1	5.5238×10^1
c_8	-2.6307	1.3947	-3.2796	1.3740
χ^2	17.4500	-	16.9363	-
R^2	0.9602	-	0.9898	-

that $t_{\text{exp}} = 1\,\text{ms}$ and $A_{\text{ISO}} = 100$ leads to $\widehat{\text{MTF}} = 1.0$ and $\hat{\sigma} = 1\,\text{DN}$.

The corresponding IOPCs as a function of the camera parameters are provided in Fig. 20. We can see that these IOPCs match well with those in Fig. 15. This is also reflected by the goodness-of-fit (χ^2) [62, pp. 595–596] and coefficient of determination (R^2) [62, pp. 484–485] statistics in Tab. IV. The omission of any polynomial term leads to significant degradations of these scores. Still, the non-monotonic part of the YOLOv4 curve could be improved, but at the expense of higher degree factors in (14).

REFERENCES

[1] J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin, "Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 14 413–14 423, 2020.

[2] V. A. Jorge, R. Granada, R. G. Maidana, D. A. Jurak, G. Heck, A. P. Negreiros, D. H. Dos Santos, L. M. Gonçalves, and A. M. Amory, "A survey on unmanned surface vehicles for disaster robotics: Main challenges and directions," *Sensors*, vol. 19, no. 3, p. 702, 2019.

[3] S. Sarkar, M. W. Totaro, and K. Elgazzar, "Intelligent drone-based surveillance: application to parking lot monitoring and detection," in *Unmanned Syst. Technol. XXI*, vol. 11021. SPIE, 2019, pp. 13–19.

[4] G. Fragapane, R. De Koster, F. Sgarbossa, and J. O. Strandhagen, "Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda," *Eur. J. Operational Research*, vol. 294, no. 2, pp. 405–426, 2021.

[5] Q. Lu, W. Zhou, L. Fang, and H. Li, "Robust blur kernel estimation for license plate images from fast moving vehicles," *IEEE Trans. Image Process.*, vol. 25, no. 5, pp. 2311–2323, 2016.

[6] I. Keller and K. S. Lohan, "On the illumination influence for object learning on robot companions," *Frontiers in Robotics and AI*, vol. 6, p. 154, 2020.

[7] S. Kühn, A. Pandey, A. Zippelius, K. Schneider, H. Erdogan, and G. Elger, "Analysis of package design of optic modules for automotive cameras to realize reliable image sharpness," in *IEEE Electr. System-Integration Technol. Conf.*, 2020.

[8] Y. Zhang, A. Carballo, H. Yang, and K. Takeda, "Perception and sensing for autonomous vehicles under adverse weather conditions: A survey," *ISPRS J. Photogramm. Remote Sens.*, vol. 196, pp. 146–177, 2023.

[9] M. Veres and M. Moussa, "Deep learning for intelligent transportation systems: A survey of emerging trends," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 8, pp. 3152–3168, 2019.

[10] S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah, "A survey of deep learning applications to autonomous vehicle control," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 2, pp. 712–733, 2020.

[11] M. A. Rahman, M. A. Rahim, M. M. Rahman, N. Moustafa, I. Razzak, T. Ahmad, and M. N. Patwary, "A secure and intelligent framework for vehicle health monitoring exploiting big-data analytics," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 10, pp. 19 727–19 742, 2022.

[12] H. Lu, H. Zhang, S. Yang, and Z. Zheng, "Camera parameters auto-adjusting technique for robust robot vision," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2010, pp. 1518–1523.

[13] I. Shim, T.-H. Oh, J.-Y. Lee, J. Choi, D.-G. Choi, and I. S. Kweon, "Gradient-based camera exposure control for outdoor mobile platforms," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 6, 2018.

[14] U. Shin, J. Park, G. Shim, F. Rameau, and I. S. Kweon, "Camera exposure control for robust robot vision with noise-aware image quality assessment," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2019.

[15] Y.-Q. Liu, X. Du, H.-L. Shen, and S.-J. Chen, "Estimating generalized gaussian blur kernels for out-of-focus image deblurring," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 3, 2020.

[16] A. Mehra, M. Mandal, P. Narang, and V. Chamola, "ReViewNet: A fast and resource optimized network for enabling safe autonomous driving in hazy weather conditions," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4256–4266, 2020.

[17] J. Aloimonos, I. Weiss, and A. Bandyopadhyay, "Active vision," *Int. J. Comput. Vis.*, vol. 1, no. 4, pp. 333–356, 1988.

[18] V. Murino, G. L. Foresti, and C. S. Regazzoni, "Adaptive camera regulation for investigation of real scenes," *IEEE Trans. Ind. Electron.*, vol. 43, no. 5, pp. 588–600, 1996.

[19] K. V. Chandrasekhar, M. H. Imtiaz, and E. Sazonov, "Motion-adaptive image capture in a body-worn wearable sensor," in *IEEE Sensors*, 2018.

[20] T. Hamamoto and K. Aizawa, "A computational image sensor with adaptive pixel-based integration time," *IEEE J. Solid-State Circuits*, vol. 36, no. 4, pp. 580–585, 2001.

[21] T. Li, Y. Song, and T. Mei, "An auto exposure control algorithm based on lane recognition for on-board camera," in *IEEE Intell. Vehicles Symp.*, 2015, pp. 851–856.

[22] J. Torres and J. M. Menéndez, "Optimal camera exposure for video surveillance systems by predictive control of shutter speed, aperture, and gain," in *Real-Time Image and Video Processing*, vol. 9400, 2015.

[23] E. Onzon, F. Mannan, and F. Heide, "Neural auto-exposure for high-dynamic range object detection," in *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2021, pp. 7710–7720.

[24] M. Kettelgerdes, L. Böhm, and G. Elger, "Correlating intrinsic parameters and sharpness for condition monitoring of automotive imaging sensors," in *IEEE Int. Conf. Syst. Reliability and Safety*, 2021.

[25] J. Xu, H. Li, Z. Liang, D. Zhang, and L. Zhang, "Real-world noisy image denoising: A new benchmark," *arXiv:1804.02603*, 2018.

[26] J. Anaya and A. Barbu, "Renoir-a benchmark dataset for real noise reduction evaluation," *J. Visual Comm. Image Repres.*, 2018.

[27] Y.-W. Tai and S. Lin, "Motion-aware noise filtering for deblurring of noisy and blurry images," in *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2012, pp. 17–24.

[28] F. Wang, L. Han, and A. J. Theuwissen, "Development and evaluation of a highly linear cmos image sensor with a digitally assisted linearity calibration," *IEEE J. Solid-State Circuits*, vol. 53, no. 10, pp. 2970–2981, 2018.

[29] J. Igual, "Photographic noise performance measures based on raw files analysis of consumer cameras," *Electronics*, vol. 8, no. 11, 2019.

[30] M. Konnik and J. Welsh, "High-level numerical simulations of noise in CCD and CMOS photosensors: review and tutorial," *arXiv*, 2014.

[31] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv:2004.10934*, 2020.

[32] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 7464–7475.

[33] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *Advances in Neural Information Processing Systems*, vol. 28, pp. 91–99, 2015.

[34] J. Cartucho, R. Ventura, and M. Veloso, "Robust object recognition through symbiotic deep learning in mobile robots," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2018, pp. 2336–2341.

[35] Y. Bai, G. Cheung, X. Liu, and W. Gao, "Graph-based blind image deblurring from a single photograph," *IEEE Trans. Image Process.*, vol. 28, no. 3, pp. 1404–1418, 2018.

[36] F. Wen, R. Ying, Y. Liu, P. Liu, and T.-K. Truong, "A simple local minimal intensity prior and an improved algorithm for blind image deblurring," *IEEE Trans. Circuits Syst. Video Technol.*, 2020.

[37] BYchao100, "Graph-based-blind-image-deblurring," https://github.com/BYchao100/Graph-Based-Blind-Image-Deblurring, 2018.

[38] FWen, "deblur-pmp," https://github.com/FWen/deblur-pmp, 2019.

[39] M. Bauer, V. Volchkov, M. Hirsch, and B. Schcölkopf, "Automatic estimation of modulation transfer functions," in *IEEE Int. Conf. Comput. Photography (ICCP)*, 2018.

[40] D.-H. Shin, R.-H. Park, S. Yang, and J.-H. Jung, "Block-based noise estimation using Adaptive Gaussian Filtering," *IEEE Trans. Consumer Electronics*, vol. 51, no. 1, pp. 218–226, 2005.

[41] G. Chen, F. Zhu, and P. Ann Heng, "An efficient statistical method for image noise level estimation," in *Int. Conf. Comput. Vis. (ICCV)*, 2015.

[42] Z. Yue, "Noise level estimation for signal image," https://github.com/zsyOAOA/noise_est_ICCV2015, 2019.

[43] H. Tan, H. Xiao, S. Lai, Y. Liu, and M. Zhang, "Pixelwise estimation of signal-dependent image noise using deep residual learning," *Computational intelligence and neuroscience*, vol. 2019, 2019.

[44] H. Tan, "Pixel-wise-estimation-of-signal-dependent-image-noise," https://github.com/TomHeaven/Pixel-wise-Estimation-of-Signal-Dependent-Image-Noise-using-Deep-Residual-Learning, 2018.

[45] K. Ma, Z. Duanmu, Q. Wu, Z. Wang, H. Yong, H. Li, and L. Zhang, "Waterloo exploration database: New challenges for image quality assessment models," *IEEE Trans. Image Process.*, vol. 26, no. 2, pp. 1004–1016, 2016.

[46] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2012.

[47] Udacity, https://github.com/udacity/self-driving-car, 2016.

[48] P. Irmisch, D. Baumbach, I. Ernst, and A. Börner, "Simulation framework for a visual-inertial navigation system," in *IEEE Int. Conf. Image Process. (ICIP)*, 2019, pp. 1995–1999.

[49] G. Singh, S. Akrigg, M. Di Maio, V. Fontana, R. J. Alitappeh, S. Khan, S. Saha, K. Jeddisaravi, F. Yousefi, J. Culley *et al.*, "Road: The road event awareness dataset for autonomous driving," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 1, pp. 1036–1054, 2022.

[50] G. Neuhold, T. Ollmann, S. Rota Bulo, and P. Kontschieder, "The mapillary vistas dataset for semantic understanding of street scenes," in *Int. Conf. Comput. Vis. (ICCV)*, 2017, pp. 4990–4999.

[51] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," in *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2020, pp. 2636–2645.

[52] H. Yin and C. Berger, "When to use what data set for your self-driving car algorithm: An overview of publicly available driving datasets," in *IEEE Int. Conf. Intell. Transp. Syst. (ITSC)*, 2017, pp. 1–8.

[53] L. Borodenko, https://github.com/LeviBorodenko/motionblur, 2020.

[54] H. Meißner, "Determination and improvement of spatial resolution obtained by optical remote sensing systems," Ph.D. dissertation, Humboldt-Universität zu Berlin, 2021.

[55] J. Pan, D. Sun, H. Pfister, and M.-H. Yang, "Blind image deblurring using dark channel prior," in *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2016, pp. 1628–1636.

[56] A. V. T. GmbH, "Prosilica gc1380h camera and driver attributes," https://cdn.alliedvision.com/fileadmin/content/documents/products/cameras/various/features/Camera_and_Driver_Attributes.pdf, 2021.

[57] I. Dr Robot, "Jaguar 4x4 wheel specification," http://jaguar.drrobot.com/specification_4x4w.asp, 2001 – 2021.

[58] Börner *et al.*, "Sensor artificial intelligence and its application to space systems–a white paper," *arXiv:2006.08368*, 2020.

[59] S. Jayaraman, S. Esakkirajan, and T. Veerakumar, *Digital Image Processing*. Tata McGraw Hill Education, 2009.

[60] S. Ray, *Applied Photographic Optics*. Focal Press, 2002.

[61] A. E. Savakis and H. J. Trussell, "On the accuracy of psf representation in image restoration," *IEEE Trans. Image Process.*, vol. 2, no. 2, pp. 252–259, 1993.

[62] J. L. Devore, *Probability and Statistics for Engineering and the Sciences*, 8th ed. Brooks/Cole, 2011.

[63] J. Janesick, *Scientific charge-coupled devices*. SPIE press, 2001, vol. 83.

[64] D. Dussault and P. Hoess, "Noise performance comparison of ICCD with CCD and EMCCD cameras," in *Infrared Systems and Photoelectronic Tech.*, vol. 5563. Int. Society for Optics and Photonics, 2004.

[65] B. Jähne and H. Haußecker, Eds., *Computer Vision and Applications*. Academic Press, 2000.

[66] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Eur. Conf. Comput. Vis. (ECCV)*, 2014, pp. 740–755.

Maik Wischow is a PhD student at the German Aerospace Center (DLR) and the Technische Universität Berlin, where he also received the M.Sc. degree in Computer Science in 2019. His research interests include computer vision, deep neural networks, optical sensor systems, optics, stereo image processing and sensor fusion.

Guillermo Gallego (SM'19) is Associate Professor at the Technische Universität Berlin, in the Dept. of Electrical Engineering and Computer Science, and at the Einstein Center Digital Future, both in Berlin, Germany. He received the PhD degree in Electrical and Computer Engineering from the Georgia Institute of Technology, USA, in 2011, supported by a Fulbright Scholarship. From 2011 to 2014 he was a Marie Curie researcher with Universidad Politecnica de Madrid, Spain, and from 2014 to 2019 he was a postdoctoral researcher at the Robotics and Perception Group, University of Zurich, Switzerland. His research interests include robotics, computer vision, signal processing, optimization and geometry.

Ines Ernst received the diploma degree in mathematics from the Technische Universität Dresden. Until 2001 she worked at the Institute for Computer Architecture and Software Technology of the German National Research Center for Information Technology (GMD FIRST). She joined the German Aerospace Center (DLR) in 2002 and is currently a research associate at the Institute of Optical Sensor Systems. Her research interests include computer vision, stereo image processing and 3D reconstruction, deep neural networks, hardware acceleration.

Anko Börner received the degree in electrical engineering from the University of Ilmenau, and the PhD degree from the German Aerospace Center (DLR). He was a Postdoctoral Researcher with the University of Zurich. He is currently the Head of the Real-Time Data Processing Department, DLR. He is author of various SCI, EI, and Scopus indexed journals and international conferences. His research interests include stereo image processing, deep neural networks, simultaneous localization and modeling (SLAM), and 3D reconstruction.