Camera Condition Monitoring and Readjustment by means of Noise and Blur

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 because they are the predominant sensors to acquire information about the environment and support actions. A camera must maintain proper functionality and take automatic countermeasures if necessary. However, there is little work that examines the practical use of a general condition monitoring approach for cameras and designs countermeasures in the context of an envisaged high-level application. We propose a generic and interpretable 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 (defocus blur, motion blur, different noise phenomena and most common combinations) by comparing traditional and retrained machine learning-based approaches in extensive experiments. Furthermore, we demonstrate how one can adjust the camera parameters to achieve optimal whole-system capability based on experimental (non-linear and non-monotonic) input-output performance curves, using object detection, motion blur and sensor noise as examples. Our framework not only provides a practical ready-to-use solution to evaluate 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.**

I. Introduction

Machines from different fields (e.g., vehicles, robots) are evolving away from manual control and gaining autonomy. This evolution is equally increasing the need for reliability and robustness to ensure the safety of people and the machines themselves. These requirements run like a red thread through all system components, starting with the sensors that perceive the environment. Particular attention is paid to trustworthy perception, as all subsequent actions depend on it. Cameras are nowadays the predominant sensors to perceive the environment, and are therefore the subject of our study. To guarantee a camera's intended functionality, autonomy also 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., [1]–[6]) have approached this task by estimating 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-white-balance, etc.). However, such techniques are typically decoupled from the envisaged high-level application and hence may not reach optimal whole-system performance. This is particularly true if the system can trade off image quality for other high-level application benefits.

This work tackles such a problem by proposing a modular and general self-health-maintenance framework that strives for optimal application performance. We demonstrate the working principle of our framework on the exemplary application of object detection (as a representative modern image 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 testing of multiple components with less effort than 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 tasks coupled to arbitrary high-level applications in order to reach optimal whole-system performance (Sec. III), and we demonstrate it for object detection affected by motion blur and noise (Sec. VII).
- We carry out a comprehensive experimental study comparing six traditional and modern (machine-learning–based, ML) image blur and noise estimators (Sec. V) on three datasets. And we provide practical recommendations with regard to camera monitoring applications (Sec. VI), all grounded on knowledge of the camera physics. In the image formation process, we account for blur (with two root causes), noise (with three root causes), and at the application level we test on two advanced object detectors (YOLOv4, Faster R-CNN) and two types of objects (cars and pedestrians).
- We provide the source code of our experiments, including the retrained estimators: https://github.com/MaikWischow/Camera-Condition-Monitoring.

II. Related Work

Our study is closely related to *active vision* [7] and *adaptive camera regulation* [8] in that there are two connected tasks: online *perception* of the current vision state and execution of

[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.

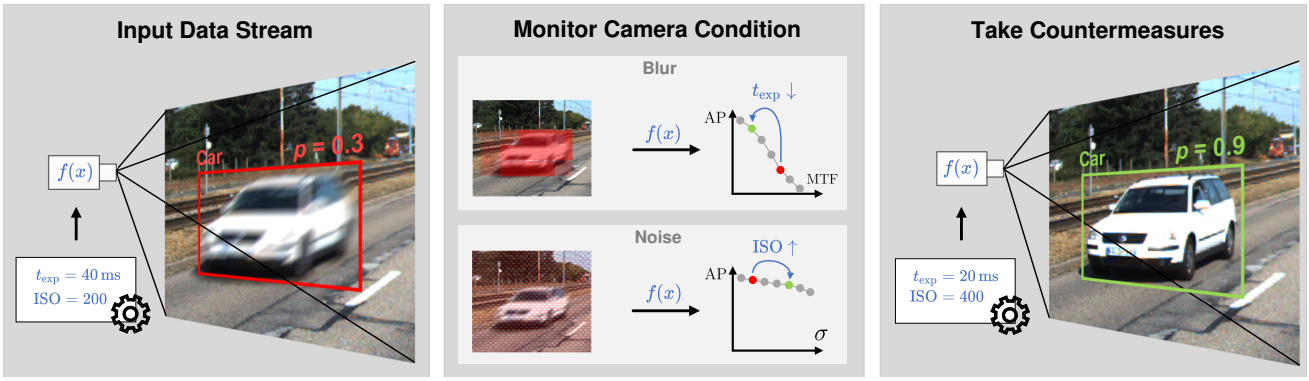


Fig. 1: The detection of the motion blurred car (red) fails with the present camera configuration (⚙). We tackle the source of this problem using (i) an online estimation of image quality properties, (ii) knowledge about camera physics $f(x)$ and (iii) empirical object detection performance curves AP (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.

an *action* to improve some target criterion. In the perception 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 [9] or a self-designed sensor [10], but it is typically managed by (automatic) exposure control through image processing. Most general exposure control works focus on image quality indicators like the intensity entropy [1], [4], gradients [2]–[4] and histograms [5], [6], or approach it learning-based [11]. Our work is most similar to [11], thus we use it as a comparison baseline. Similarly to [4] and [11], we employ blur and noise as image quality indicators for the camera's condition. The work in [4] assumes a simplified additive zero-mean Gaussian noise, while [11] models the most important noise sources close to the camera physics (which is experimentally supported as being more realistic [12], [13]). In contrast to all aforementioned works, we consider a more extensive and realistic image formation pipeline by including motion and defocus blur as well as simultaneously occurring corruptions that influence each other.

Traditional works typically do not couple the estimation problem they solve to the envisaged high-level application and hence do not reach optimal whole-system performance (e.g., [1], [3], [6]). We incorporate object detection performance as a feedback signal [11], but we do not rely on a tailored end-to-end learning approach. Instead, we propose a more modular, extensible and interpretable approach: 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 [2]–[4], [11] to focus on regions of interest. The details of these estimators and the adaptations made are presented in upcoming sections (Secs. V-A and V-B).

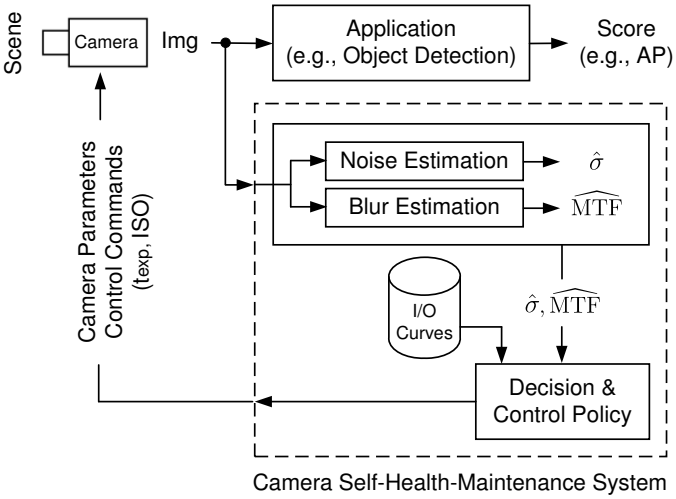


Fig. 2: *System Overview*. The camera is constantly monitored by analyzing image corruptions (e.g., blur and noise). According to the estimated severity of such corruptions, camera control parameters (e.g., exposure time t_{exp} and ISO gain) are recalculated to maximize application performance using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

III. System 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. 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 (Sec. IV). The pipeline contains the most common (physics-based) sources of blur and noise affecting the camera condition, with realistic severity levels (Sec. VI-A). We quantify these levels using noise and blur

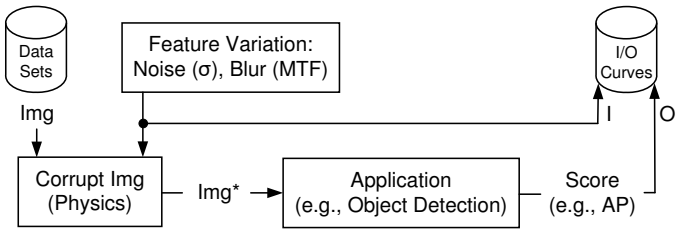


Fig. 3: *Training the System (System Identification)*. An offline sensitivity analysis determines the impact of physical image corruptions (e.g., blur and noise) on the performance of a target application (e.g., object detection), and stores the results in input/output (I/O) performance curves. As input,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

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 objective *average precision score* (AP, Sec. V-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 machine learning based, real-time capable noise level and MTF estimators (Sec. V). We have evaluated their estimation and runtime performances compared to established state-of-the-art estimators (Sec. VI) for isolated and combined corruption cases, and propose a simple approach to improve blur estimation in case of interfering high noise levels. 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. VII).

IV. 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 [14, 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:

$$I^*(x,y) = I(x,y) \circledast h(x,y), \tag{1}$$

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) [14, 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:

$$\mathrm{PSF}(x,y) \quad \overset{\mathcal{F}}{\mapsto} \quad \mathrm{OTF}(f) \propto \mathrm{MTF}(f)\, e^{i\,\mathrm{PhTF}(f)}. \tag{2}$$

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)) [14, p. 325]:

$$h(x,y) = \begin{cases} s, & (x-c_x)^2 + (y-c_y)^2 \leq r^2 \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

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

$$d = A\,\frac{\mathrm{f}}{S_1 - \mathrm{f}}\,\frac{|S_2 - S_1|}{S_2}, \tag{4}$$

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) [15, p. 216]. 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 [14, 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 $h(x,y)$ 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 [14, 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

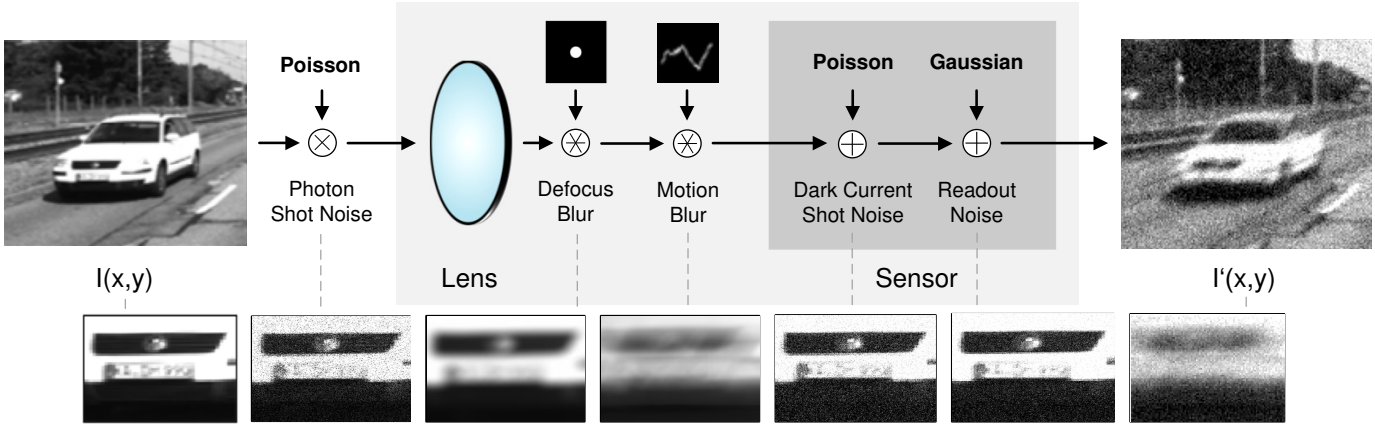


Fig. 4: *Image formation process* of the considered camera system, including blur and noise models. A clean image $I(x, y)$ undergoes several physical processes that produce noise and blur, yielding the corrupted image $I'(x, y)$ (clean image patch vs. distinct corruptions in stated order). Noise is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

residuals are assumed to have a minor influence on image quality. Generally, noise can be modelled by:

$$\tilde{I}(x, y) = I(x, y) + I(x, y)^{\gamma}\, u(x, y), \tag{5}$$

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 noisy 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 [16].

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 [17, p. 225]:

$$\mathcal{P}_{\lambda}(k) = \frac{\lambda^k}{k!} e^{-\lambda} \overset{k \to \infty}{\approx} \frac{1}{\sqrt{2\pi\lambda}}\, e^{-(k-\lambda)^2/2\lambda}, \tag{6}$$

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 (5) 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 (6). 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 [18, ch. 7.1.1]. DCSN is signal-independent, hence $\gamma = 0$ in (5).

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 [19, 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 [18, 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 (5). We keep it at this level of abstraction and refer to [16] for details.

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

V. Image Quality Assessment

In this section we introduce blur and noise estimation methods to quantify image quality objectively. We summarize both traditional and learning-based (ML) approaches first and then detail our improvements (Secs. V-A and V-B). Subsequently, we introduce object detection as a proxy application to assess the severity of blur and noise levels (Sec. V-C).

A. Blur Estimation (via the MTF)

The goal of our image blur estimators is to predict the MTF given a possibly blurred input image patch (1) I^*, where I^* is assumed to be monochrome (i.e., grayscale) and of size 192×192 pixels (following the ML approach). Fig. 5 summarizes the steps of the two main approaches.

1) Traditional methods (non-learning–based): We use two baseline methods: "*graph-based*" [20] (GBB) and "*simple local minimal intensity prior*" [21] (PMP) as traditional blur kernel estimators (top branch in Fig. 5). Both estimators follow a maximum-a-posteriori framework

$$\min_{I,h} \mathcal{L}(I \circledast h, I^*) + \alpha G(I) + \beta R(h) \tag{7}$$

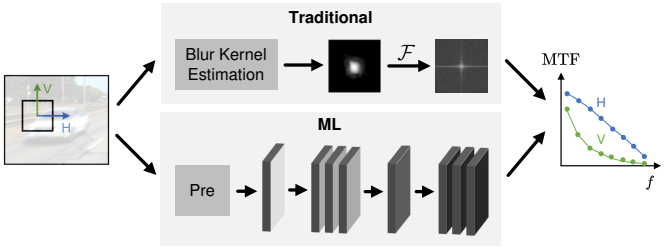


Fig. 5: *Blur estimation* of traditional (top branch) and learning-based (bottom branch, ML) approaches. All methods take one or more image patches as input and output estimated MTF samples for pre-defined image *frequencies* (f) in the *horizontal* (H) and *vertical* (V) directions. Traditional methods first estimate a blur kernel, transform it into 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.

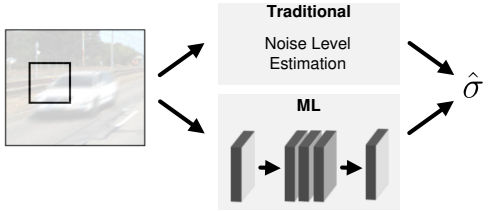


Fig. 6: *Noise estimation* of traditional or learning-based (ML, e.g., multi-layer CNN-based) approaches. Both approaches estimate a noise level $\hat{\sigma}$ for each input image patch.

to iteratively refine a clean latent image I and the blur kernel h. The objective function (7) is the negative logarithm of the posterior distribution (thus maximization turns 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 method [20] 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 [21] 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 small computation complexity. The resulting h from each method is Fourier-transformed into the MTF (2) and sampled at the same spatial frequencies as the learning-based approach (Fig. 5), for better comparison. We use the source code from [22], [23], setting the kernel size parameter to 31×31 pixels.

2) Learning-based Method: We modify a learning-based approach [24] to *directly* estimate MTF values of camera lenses 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 for estimations of 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 downsampled to enlarge the receptive field of early CNN layers. We alter step (ii) to distinguish between estimations in horizontal and vertical directions and thus be able to compare to baseline methods GBB and PMP.

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. 5). 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 [24], pairs of sharp image patches and PSFs (I, h), synthetic or real, are collected. Their convolution (1) leads to the training samples I^*; and the respective MTF samples of the PSFs at pre-defined frequencies are the training labels. In contrast to [24], we blurred the sharp images by simulated random defocus and motion blur kernels (see Sec. VI) and retrained the CNN.

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 the authors' experiments, 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 reimplemented it with guidance from the authors, who also provided their original training data.

B. Noise Estimation

The goal of the image noise estimators is to predict the noise level σ of a process (5) u given a noisy input image patch $\tilde{I}$, which is monochrome and of size of 128×128 pixels (following the ML approach). Fig. 6 depicts the steps of the two main approaches.

1) Traditional methods (non-learning–based): As baseline estimators we use the works of [25] (self-implemented) and [26] (with its code basis [27]). Both are representatives of the two major noise estimation approaches in the literature:

The *adaptive Gaussian filtering method* [25] (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 due to better observed results on the selected datasets.

The method [27] 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 [28], [29] as learning-based (ML) approach. It was designed for pixel-wise noise level estimation from signal-dependent noisy images. The noise model was Gaussian with parameters that accounted for photon and readout noise.

The CNN consists of 16 convolutional layers (including three residual blocks) and lacks pooling and interpolation

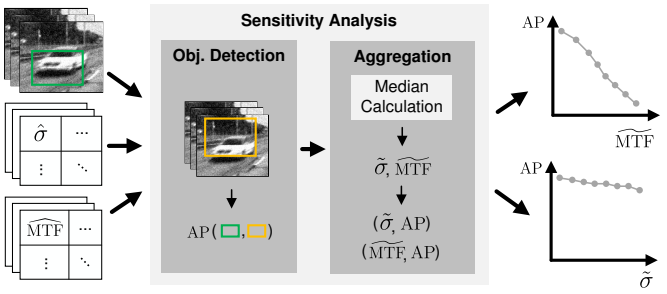


Fig. 7: *Sensitivity analysis* of object detector performances for blur and noise. The detectors are evaluated on corrupted images resulting in *average precision* (AP) scores. For the true detection areas, corresponding patch-wise noise ($\hat{\sigma}$) and blur ($\widetilde{\text{MTF}}$) estimations are aggregated to medians ($\tilde{\sigma}$ and $\widetilde{\text{MTF}}$) and, together with the APs, added to performance curves.

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

layers due to a known performance decrease for image noise tasks. The resulting output $\hat{\sigma}$ is estimated for each pixel, but for a better comparison with baseline methods we use the median over the patch as the noise level estimator.

Training in [28] is supervised, carried out by artificially adding noise with $\sigma \in [0, 30]$ to images from the Waterloo dataset [30]. We retrained the CNN in the same way using our noise model of Sec. IV-B (see details in Sec. VI).

C. Empirical Input-Output Performance Curves

We choose object detection as a representative modern image application of great importance in various fields. And, as presented, we choose image noise and blur as the main data quality indicators of the state of our imaging system.

Specifically, we use YOLOv4 [31] and Faster R-CNN [32] as state-of-the-art real-time object detectors (with pre-trained models and default settings, applied on grayscale images). The goal is to determine object detection sensitivities *empirically* for different noise and blur levels (Fig. 7). We do this analysis offline, but an iterative online approach is also feasible.

As input we assume image data with ground truth object detections and corresponding patch-wise blur ($\widetilde{\text{MTF}}$) and noise ($\hat{\sigma}$) estimations (Fig. 7). First, both object detectors are applied on the images to gather estimated detections. Second, these estimations are scored with the well-known average precision (AP) metric, which we calculate following [33]. In the subsequent aggregation step, we determine the corresponding median $\tilde{\sigma}$ and $\widetilde{\text{MTF}}$ estimations of all the respective image patches overlapping with the ground truth object detections. Finally, the resulting input-output tuples ($\tilde{\sigma}$, AP), ($\widetilde{\text{MTF}}$, AP) or ($\tilde{\sigma}$, $\widetilde{\text{MTF}}$, AP) are collected as *performance curves* (IOPCs).

VI. Experiments

We first describe the datasets used and the image corruptions applied (Sec. VI-A). Subsequently, we evaluate the accuracy and runtime performances for the proposed blur and noise estimators separately (Secs. VI-B and VI-C) and on combined image corruptions (Sec. VI-D). Finally, we propose a post-processing technique to enable blur estimation in the presence of high noise levels (Sec. VI-E). 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 [34] and Udacity [35] (Fig. 8). We create *Sim* with the simulator [36] 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.

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

Noise: Following the "real noise" studies in [13], 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 [16] and study noise in isolation or in combination. (*i*) For isolated DCSN and readout noise studies, we set the temperature to $T = 330\,\text{K}$ and the exposure time to $t_{\text{exp}} = 0.1\,\text{s}$. (*ii*) For the combined noise case we include *all* noise sources, with random $T \in [300, 330]\,\text{K}$ and $t_{\text{exp}} \in [0.002, 1]\,\text{s}$ 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 (inspired by [37]). Defocus blur kernels are calculated analytically using (3). Motion blur kernels are generated using [38], 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 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, 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):

$$\begin{aligned} \text{AMAE} &\doteq \frac{1}{2} \sum_{w=\{\text{H,V}\}} \text{MAE}(w), \\ \text{MAE}\,(w) &\doteq \frac{1}{8} \sum_{i=1}^{8} \left| \text{MTF}_w^{\text{GT}}(f_i) - \widehat{\text{MTF}}_w(f_i) \right|. \end{aligned} \tag{8}$$

We first calculated (robust) median, minimum and maximum estimations for the uncorrupted datasets in Fig. 9. In the *Sim* case, we could determine MTF^{GT} by evaluating a Siemens Star (generated in the simulator) with the tool [39]. 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 GBB/PMP —unlike the CNN— usually perform worst for small ($3\,\text{px}$) and large ($21\,\text{px}$) kernel sizes. 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-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 [20] 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, whereby the impact decreases towards larger kernels. 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, with a median AMAE of $< 1\,\%$. 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. It is conspicuous that a kernel size $d = 15\,\text{px}$ falls out of the grid for *Sim*. We notice that all methods have almost no errors in horizontal but large errors in vertical direction (above all, the CNN method). This is due to the already mentioned lack of strong horizontal edges of the evaluated image patches, which are necessary for larger kernel sizes, hence we ignore these results.

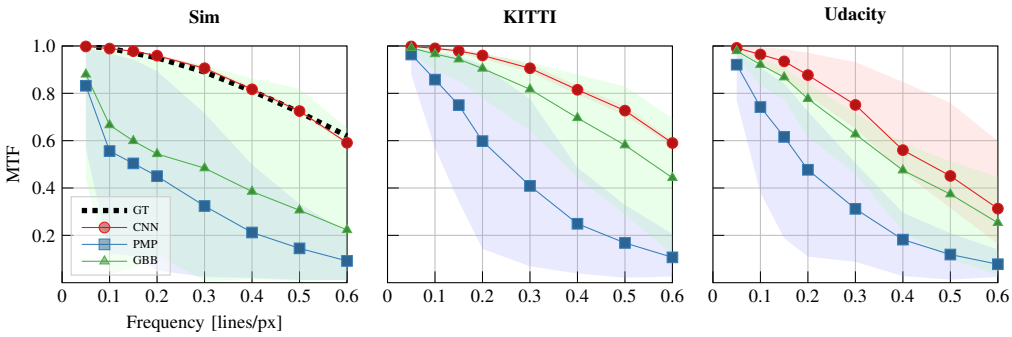
Non-linear motion estimation results (also in Tab. I) differ for the CNN method, which produces larger and more varying errors for kernel sizes $d \geq 7\,\text{px}$ compared to the traditional estimators and the linear case. We interpret this as a larger uncertainty and conclude that the CNN might not be appropriate for estimation of complex non-linear motion kernels. In contrast, the scores of GBB/PMP are more accurate and more stable 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 [20], where PMP is compared to the work of Pan et al. [40] 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. Although the CNN 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 accurate estimates for medium-sized kernels and in the case of non-linear motion blur. The CNN method, on the other hand, might not be suited for 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 (re-



Mean Runtime [s]	
CNN_{MTF}	0.24
PMP	12.69
GBB	13.07
CNN_{σ}	0.005
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_{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	**0.2**	**0.7**	**0.5**	**0.3**	**0.7**	**12.2**	12.1	6.2	25.1	**3.8**	**2.8**	18.2	22.8	12.2	32.0
	PMP	7.9	2.2	2.4	2.4	2.6	34.7	5.5	12.5	13.3	19.4	25.8	4.7	**5.2**	**3.6**	11.7
	GBB	2.9	3.8	4.3	4.8	9.1	25.0	**5.4**	**4.7**	17.0	11.1	14.8	**4.6**	6.2	6.0	**7.2**
KITTI	CNN	**0.4**	**0.8**	**0.5**	**0.3**	**0.5**	**3.1**	11.1	**4.4**	8.0	**7.3**	**2.8**	15.5	16.5	4.7	28.7
	PMP	9.2	2.3	2.3	2.0	2.1	35.1	6.6	14.4	**3.1**	20.3	24.6	4.9	6.9	**2.8**	9.1
	GBB	3.7	2.4	3.1	4.5	10.1	17.8	**4.6**	8.4	5.8	16.7	11.3	**4.7**	**5.8**	5.2	**8.3**
Udacity	CNN	**3.8**	**0.6**	**0.6**	**0.3**	**0.5**	**15.7**	7.6	**6.0**	**4.5**	**10.8**	**12.4**	9.5	12.5	5.0	12.6
	PMP	14.3	4.3	3.6	2.2	1.8	39.5	9.2	17.4	5.0	22.9	30.0	8.8	10.5	**4.2**	10.8
	GBB	6.8	2.9	3.2	4.5	11.3	24.4	**6.2**	10.6	5.0	19.1	16.5	**5.1**	**6.1**	5.6	**10.5**

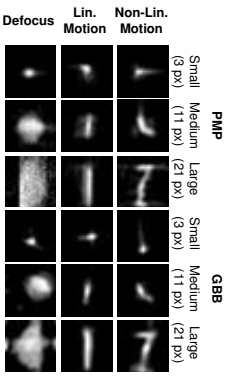


Fig. 10: *Challenging conditions in the Udacity dataset*. From top left: Slight motion blur (3 px) in moving direction, light reflections and two examples of severe motion blur.

jecting $\approx 5\,\%$ 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 [25], [26]. 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; [4] 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 $\sigma \leq 30\,\text{DN}$. 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 [12], [13], 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 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,

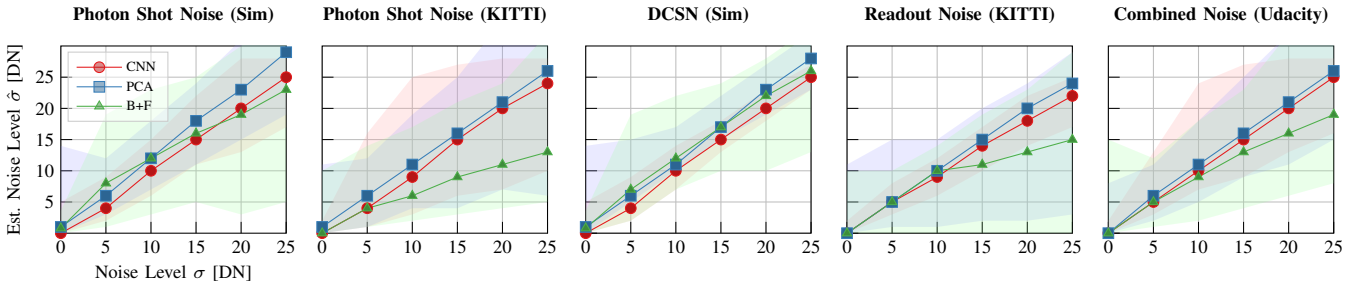


Fig. 11: *Noise estimation of corrupted datasets.* Median, minimum and maximum statistics (depicted by sampled points with interpolation in between and the shaded areas, respectively) of the three proposed noise estimators (CNN, PCA, B+F) as the noise level σ increases (from 0 to 25 grayscale levels, 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).

similar results have been shown by comparing traditional and learning-based methods on real data [12].

Computational performance: Regarding runtime (Fig. 9, right table), PCA is executed fastest, with an average of 2 ms per patch, but in the same order of magnitude as the other estimators (5 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 on Udacity ("Defocus + DCSN"), and Photon Shot Noise with simultaneous linear motion blur on KITTI ("Photon + Lin. Motion").

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.

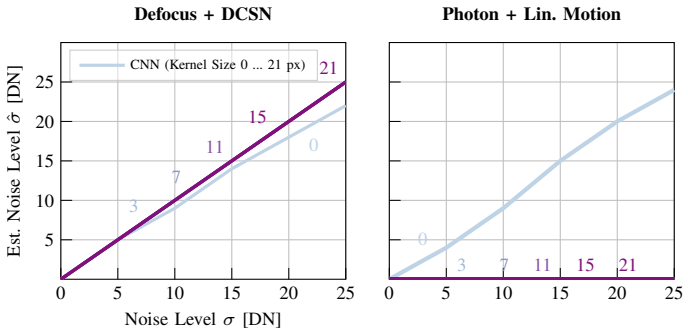
On the other hand, DCSN negatively affects defocus estimation because advantageous information for detecting blur (the absence of high frequencies) gets corrupted by noise. We notice two effects from the results on the table of Fig. 12: All defocus estimations worsen with increasing noise levels, and this impact becomes more severe for larger kernel sizes. While estimations for a small kernel ($d = 3\,\mathrm{px}$) can be considered as still good for $\sigma = 10\,\mathrm{DN}$, the same noise level leads to poor blur estimations for larger kernels. This outcome was investigated in the context of motion deblurring [41], 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 robust for $\sigma \leq 5\,\mathrm{DN}$. 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. As we see on the second plot and the table in Fig. 12, the results meet the expectations. A motion blur of size $d = 3\,\mathrm{px}$ is already enough to fully disturb noise estimation, resulting in a constant $\hat{\sigma} = 0$ (note that noise is not removed from the image but spread among neighboring pixels). The linear motion blur estimations remain untouched among all blur and noise levels. The CNN blur estimator seems to focus on high-frequency image content as the results from Defocus + DCSN already suggested.

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, photon noise does not harm, while subsequent DCSN with $\sigma \geq 10\,\mathrm{DN}$ generally prevents estimation. Hence, if one can eliminate photon noise, we suggest estimating noise before judging a blur estimation result. As in the noise evaluation of Sec. VI-C, sensor noise (DCSN and readout noise) is more favorable than photon shot noise for a condition monitoring application.

E. Improved Blur Estimation in Presence of High Noise

The previous section 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. The approach exploits the fact that preceding photon noise does not influence the MTF estimation of subsequent linear motion blur (see table in Fig. 12). Hence, the approach consists



		Defocus + DCSN					Photon + Lin. Motion				
Size [px]		3	7	11	15	21	3	7	11	15	21
Kernel		▪	●	●	●	●	▬	▞	▮	▰	▮
Noise Level	0	3.8	0.6	0.6	0.3	0.5	3.1	11.1	4.4	8.0	7.3
	5	4.3	4.9	1.7	0.8	1.1	3.1	11.1	4.4	8.0	7.3
	10	6.0	33.7	37.4	48.3	39.4	3.1	11.1	4.4	8.0	7.3
	15	19.0	49.4	61.4	69.6	76.3	3.1	11.1	4.4	8.0	7.3
	20	24.0	50.3	62.2	70.0	76.5	3.1	11.1	4.4	8.0	7.3
	25	24.0	50.4	62.3	70.1	76.7	3.1	11.1	4.4	8.0	7.3

Fig. 12: *Combined estimations of blur and noise* for two image corruption configurations: Defocus + DCSN on the Udacity dataset, and Photon + Linear Motion blur on the KITTI dataset. Plots of the median noise estimation (Left) and table with median blur estimation (AMAE (8) 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.

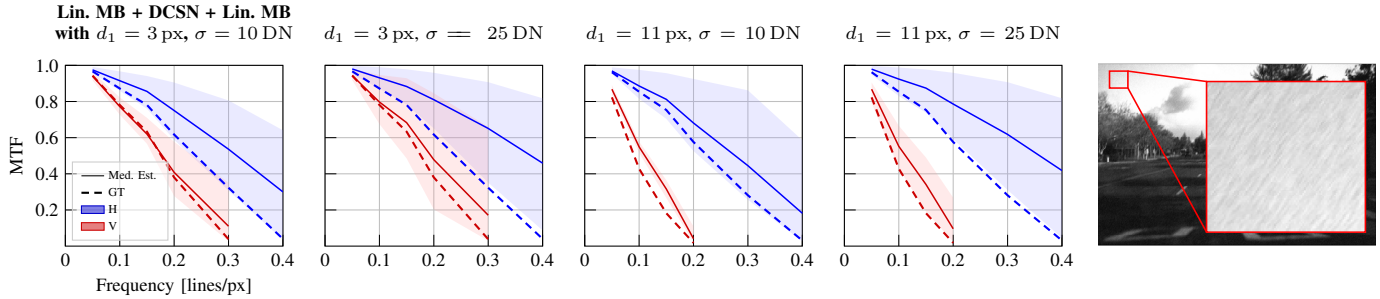


Fig. 13: *Improved blur estimation in presence of noise.* Scenario "lin. MB_1 + DCSN + lin. MB_2", with kernel size $d_2 = 7\,\text{px}$. Left: four plots of min./max./median MTF estimations (Est.) of the combined linear motion blur (lin. MB 1 and 2) compared to the ideal combined MTF (GT) in horizontal (H) and vertical (V) image directions on the Udacity dataset. The last motion blur (lin. MB_2) spreads DCSN and thus fixes MTF estimation for the combined blur. However, noise might become structured (Right) and influence median and max. estimations; minimum estimations stay stable and approach the ground truth best.

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. VI-D, the considered pipeline has now three stages: lin. MB + DCSN + filtering. Letting the first blur kernel be b_1, we filter noise by an additional kernel b_2, estimate the overall blur $\widehat{\text{MTF}}(b_1, b_2) = \widehat{\text{MTF}}(b_1)\,\widehat{\text{MTF}}(b_2)$ and lastly divide the MTF by the known $\text{MTF}^{\text{GT}}(b_2)$ according to the Fourier convolution theorem [42, p. 242]. To this end, we assume $\text{MTF}^{\text{GT}}(b_2) \approx \widehat{\text{MTF}}(b_2)$ and determine the estimation error of $\widehat{\text{MTF}}(b_1)$ with respect to $\text{MTF}^{\text{GT}}(b_1)$.

Due to the combinatorial complexity of the experimental configuration, we focus on the following one. Supported by the results in Sec. VI-D, we employ only the CNN method for MTF estimation on Udacity data, and in preparation for Sec. VII, we consider the case of lin. MB + DCSN (representative for sensor noise, to keep it clear and concise). Although subsequent defocus blur might be the best choice to filter DCSN, our CNN has not been trained on combined defocus and MB, hence we apply a lin. MB filter (which leads to the configuration lin. MB_1 + DCSN + lin. MB_2). To approach $\text{MTF}^{\text{GT}}(b_2) \approx \widehat{\text{MTF}}(b_2)$, we pick the motion blur kernel with size $d_2 = 7\,\text{px}$ (compare results in Table I). Finally, we select the four operating points with small/large kernel sizes ($d_1 \in \{3, 11\}\,\text{px}$) and high/higher noise levels ($\sigma \in \{10, 25\}\,\text{DN}$). The blur sizes are chosen so that the overall blur is still detectable by the CNN. In the following, we first evaluate the estimation of $\widehat{\text{MTF}}(b_1, b_2)$ (summarized in Fig. 13) and continue with the division by the known $\text{MTF}^{\text{GT}}(b_2)$, i.e., evaluating $\widehat{\text{MTF}}(b_1)$ (Table II).

1) $\widehat{\text{MTF}}(b_1, b_2)$ Estimation: The four plots in Fig. 13 show robust min./median/max. estimations (with $\approx 5\%$ outliers removed) compared to the respective $\text{MTF}^{\text{GT}}(b_1, b_2) = \text{MTF}^{\text{GT}}(b_1)\,\text{MTF}^{\text{GT}}(b_2)$ (calculated using the known blur kernels) in horizontal and vertical directions. Frequencies beyond the zero-crossing of the MTF^{GT} curves are omitted because they only contain noise (aliasing).

Our first observation is the growing estimation uncertainty (variance) towards higher frequencies and with higher noise levels. This effect originates from the filtered DCSN, which forms a fine-detailed structure in diagonal direction. It also influences lower frequency details and increases the median and max. estimations in both directions (see image in Fig. 13).

We further observe low estimation variances in the vertical direction if $d_1 = 11\,\text{px}$. This is because the MB_1 kernel operates mainly in the vertical direction, which supports the respective MTF estimations. Last but not least, we notice that the minimum MTF estimations approach the GT values best. These estimations originate from unaffected image structure

TABLE II: *Estimation of linear motion blur* b_1 (lin. MB) on combined pipeline (Lin. MB_1 + DCSN + Lin. MB_2), using Udacity data. For Lin. MB_2 we use a kernel size of $d_2 = 7\,\text{px}$. The table reports mean absolute errors (MAE) of horizontal (H) and vertical (V) estimations, their average (AMAE) and the expected AMAE (Eq. (9)).

Corruption Levels		Error Metrics			
d_1 [px]	σ [DN]	MAE (H)	MAE (V)	AMAE	$\text{AMAE}^{\text{Exp.}}$
3	10	2.8	11.6	7.2	17.4
3	25	3.0	21.1	12.1	17.4
11	10	6.3	8.7	7	9.7
11	25	3.3	1.6	2.45	9.7

(e.g., structure orthogonal to the noise structure direction).

2) $\widehat{MTF}(b_1)$ Estimation: We need to ensure three preconditions to divide $\text{MTF}^{\text{GT}}(b_1)$ from $\widehat{\text{MTF}}(b_1, b_2)$ for a meaningful result: (i) $\widehat{\text{MTF}}(b_1, b_2) \leq \text{MTF}^{\text{GT}}(b_1)$, (ii) $\text{MTF}^{\text{GT}}(b_1) > 0+\epsilon$ and (iii) $\widehat{\text{MTF}}(b_1, b_2) > 0+\epsilon$, for all sampled frequencies. We chose the control parameter $\epsilon = 0.1$ to avoid large quotients for small values, and omit frequencies that do not satisfy the conditions.

Table II presents results in terms of the MAE and AMAE scores (8), and the expected AMAE scores

$$\text{AMAE}^{\text{Exp.}} \doteq \sqrt{\text{AMAE}(\widehat{\text{MTF}}(b_1))^2 + \text{AMAE}(\widehat{\text{MTF}}(b_2))^2} \quad (9)$$

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

We observe noticeably worse MAE(V) scores for $d_1 = 3\,\text{px}$ than for $d_1 = 11\,\text{px}$, which are in agreement with the already-mentioned slight motion blur in the moving direction (V) on Udacity data (Sec. VI-B). The high noise level ($\sigma = 25\,\text{DN}$) further disturbs the blur estimation for the small kernels. Vice versa, the high noise seems to boost the blur estimation for $d_1 = 11\,\text{px}$. This observation is in agreement with the corresponding plots of Fig. 13: the minimum $\widehat{\text{MTF}}(b_1, b_2)$ estimations decrease slightly due to the additional noise structure but simultaneously reduce the deviations from the ground truth. Generally, we observe lower AMAE scores for all configurations than expected by the error propagation and attribute these differences to such a partial error counteraction.

Summarizing, additional filtering suppresses noise so that preceding blur estimation can be re-enabled for high sensor noise levels $\sigma \geq 10\,\text{DN}$. The estimation of the (preceding) blur even comes with a lower error than expected in all configurations. Although the procedure might come with side effects (in case of a filter that transforms noise into structured image content), they can be avoided by taking the minimum MTF estimations over time. This procedure is also suitable for a condition monitoring application as it can be applied in the background without changing the camera configuration.

VII. Maximizing Object Detection by Trading Off Blur and Noise

We now demonstrate how one can use the online blur/ noise estimators and the offline empirical input-output performance curves to control image quality and hence optimize the system's performance (Fig. 2). We choose object detection as our exemplary target application and select "car" and "pedestrian" as exemplary object classes on Udacity data. We focus on actions tackling linear motion blur (lin. MB) here because object detectors are substantially more sensitive to lin. MB than to noise (Fig. 14), and there is abundant motion blur in standard datasets like Udacity (Fig. 10). We also neglect photon noise and demonstrate the procedure with sensor noise only (DCSN + read noise), so that filtered photon noise does not lower the noise level estimation. Unless otherwise stated, we apply the settings from Sec. VI-A for noise generation.

We make the following considerations knowing the camera's physical process. The main controllable influencing factor of motion blur is the camera's exposure time t_{exp} (Sec. IV-A2). We exploit the relations

$$\begin{gathered} t_{\text{exp}} \propto I \text{ and } t_{\text{exp}} \propto \text{MB} \sim \text{MTF}^{-1} \sim \text{AP}, \\ \text{ISO} \propto I \text{ and } \text{ISO} \propto \sigma \sim \text{AP}^{-1}, \end{gathered} \quad (10)$$

where AP is the average precision of the object detector.

Reducing t_{exp} by a factor $\alpha \doteq t_{\text{exp}}^{\text{old}}/t_{\text{exp}}^{\text{new}} > 1$ equally decreases the aggregated amount of light intensity I (assuming sensor linearity) and also MB by the same factor (assuming constant relative speed between camera and scene). To compensate for the missing light, we may increase the camera ISO gain by factor α, which likewise amplifies the noise level σ. This relationship depends on the camera sensor architecture and whether the analog or digital signal is amplified [43]. 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:

$$\alpha^\star = \arg\max_{\alpha} \ \text{AP}(\alpha\,\hat{\sigma}, \alpha\,\text{MB}(\widehat{\text{MTF}})). \quad (11)$$

Note that $t_{\text{exp}} \propto \sigma^2_{DCSN}$ during optimization [16, p. 3]. We drop this influence here for simplicity, since the already small DCSN has no significant effect on $\hat{\sigma}$ in our setting.

Fig. 14 shows exemplary 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 we do not really know what machine learning methods learn.

Example 1: We 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. 15). The left image in Fig. 15 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 $t_{\text{exp}} = 28\,\text{ms}$ induces a motion blur of $d_{\text{old}} \approx 21\,\text{px}$), and we also apply sensor noise of $\sigma = 3\,\text{DN}$. In this situation (second image in Fig. 15) blur and noise are estimated within the expected error ranges ($\hat{d}_{\text{old}} \approx 18\,\text{px}$), but the cars are no longer detected ($p \approx 0$). In the next step, we

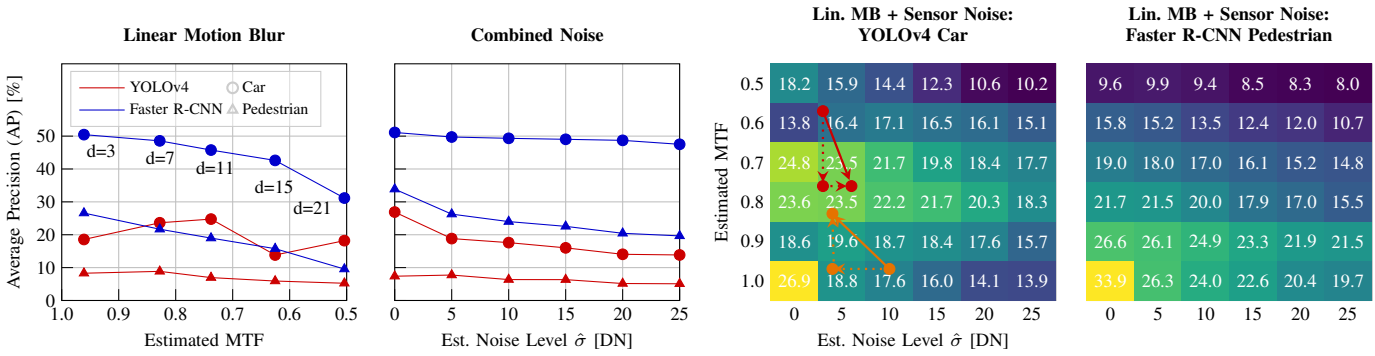


Fig. 14: *Influence of blur and noise on object detection performance.* Exemplary object detection performances depending on isolated (Left) and combined (Right) occurring and blur and noise. All input-output profiles depend on the actual estimated corruption levels. Performances are measured in terms 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. VII).

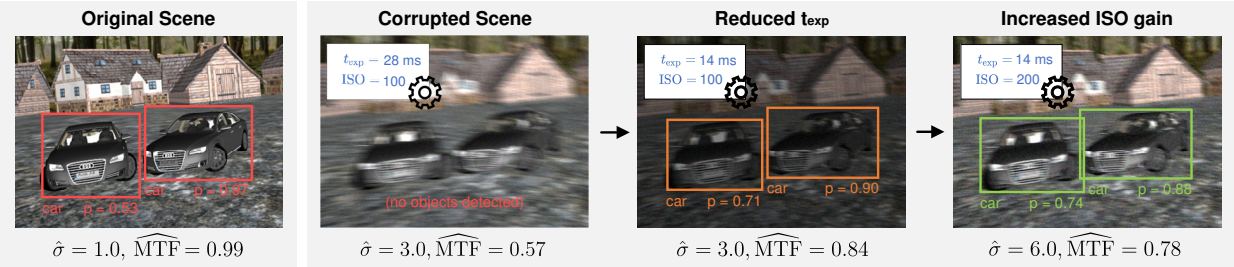


Fig. 15: *Maximizing object detection by trading off blur and noise.* Application of the proposed framework to detect cars using YOLOv4 on *Sim* data suffering from linear motion blur and sensor noise. The scene (Left) is first imaged with an exposure time of 28 ms (leading to a motion blur of $d \approx 21$ px) and corrupted by noise of $\sigma = 3$ DN. As a result, YOLOv4 no longer detects the cars (Center-Left). Applying the optimal $\alpha^\star \approx 2$ (according to the performance profile from Fig. 14) improves car detection (Center-Right), even beyond the value in the uncorrupted scene. Finally, we multiply the ISO gain by $\alpha^\star$ to compensate for the missing light, which improves overall detection slightly (Right). Hence, blur is reduced from $d \approx 21$ px to 9 px and noise amplifies from $\sigma \approx 3$ DN to 6 DN while detection rate increases from zero (no detection) to $p \approx 0.8$.

determine $\alpha^\star$: knowing the relation between motion blur sizes and estimated MTFs (first plot in Fig. 14) and the estimated noise level, we target an $\widehat{\text{MTF}} \in [0.7, 0.8]$, which corresponds to $\hat{d} \in [8, 12]$ px (cf. first heat plot in Fig. 14). We chose $d_{\text{target}} \approx 9$ px, hence, $\alpha^\star = \hat{d}_{\text{old}}/d_{\text{target}} \approx 18/9 = 2$. We then reduce t_{exp} by the factor $\alpha^\star$ and show an intermediate image without ISO gain amplification. The cars are now well detected 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 the ISO gain by the factor $\alpha^\star$ to restore the original intensity level, producing the last image of Fig. 15. In this last step the detection scores and the accuracy of the estimated blur ($\hat{d}_{\text{target}} \approx 9$ px) slightly increase despite the likewise noise amplification ($\sigma \approx \hat{\sigma} = 6$ DN). The steps taken are marked with red arrows on the heat plot in Fig. 14.

Example 2: Let us show another example on the non-monotonic YOLOv4–car-detection heat map of Fig. 14, marked with orange arrows. Suppose the initial operating point is given by an image with $(\hat{\sigma}, \widehat{\text{MTF}}) \approx (10, 0.96)$. The system decides to decrease the noise at the expense of increasing motion blur to move to higher AP detection values (greener part of the heat map). Inspecting the AP curves (1st plot in Fig. 14), $\text{MTF} = 0.96$ corresponds to $d = 3$ px, and the system targets $\widehat{\text{MTF}} \in [0.7, 0.8]$ (high values of the heat map), which corresponds to a larger motion blur of $\hat{d} \approx 8$ px. Two steps are taken (like in Fig. 15): first, the system increases the exposure time by a factor $\alpha = 8/3 \approx 2.7$ to achieve the desired MTF improvement. Then, it decreases the ISO (and reduces noise) by the same factor $\alpha \approx 2.7$ to restore the intensity level for the detector. The final operating point is $(\hat{\sigma}, \widehat{\text{MTF}}) \approx (3.8, 0.8)$, which has a higher AP value than the initial point.

VIII. 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 system 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 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; 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 influencing effects in image processing.

Lastly, we have focused on corruptions originating in the camera itself. However, a possible extension would be to model and compensate for unexpected camera conditions (i.e., corruption sources that cannot be avoided or that originate outside the camera). This could be done by acquiring and exploiting additional data, such as the sensor'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 [44].

REFERENCES

[1] 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.

[2] 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.

[3] Z. Ding, X. Chen, Z. Jiang, and C. Tan, "Adaptive exposure control for image-based visual-servo systems using local gradient information," *JOSA A*, vol. 37, no. 1, pp. 56–62, 2020.

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

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

[10] 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.

[11] E. Onzon, F. Mannan, and F. Heide, "Neural auto-exposure for high-dynamic range object detection," in *IEEE CVPR*, 2021, pp. 7710–7720.

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

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

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

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

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

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

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

[19] 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.

[20] 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.

[21] 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.

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

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

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

[25] 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.

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

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

[28] 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.

[29] 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.

[30] 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.

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

[32] 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.

[33] 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.

[34] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *IEEE CVPR*, 2012.

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

[36] 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.

[37] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman, "Understanding and evaluating blind deconvolution algorithms," in *IEEE CVPR*, 2009.

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

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

[40] J. Pan, D. Sun, H. Pfister, and M.-H. Yang, "Blind image deblurring using dark channel prior," in *IEEE CVPR*, 2016, pp. 1628–1636.

[41] Y.-W. Tai and S. Lin, "Motion-aware noise filtering for deblurring of noisy and blurry images," in *IEEE CVPR*, 2012, pp. 17–24.

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

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

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