2014 IEEE 17th International Conference on
Intelligent Transportation Systems (ITSC)
October 8-11, 2014. Qingdao, China

Detection of Camera Artifacts from Camera Images

Nils Einecke[1] and Harsh Gandhi[2] and Jörg Deigmöller[1]

Abstract— **Cameras are frequently used in state-of-the-art systems in order to get detailed information of the environment. However, when cameras are used outdoors they easily get dirty or scratches on the lens which leads to image artifacts that can deteriorate a system's performance. Little work has yet been done on how to detect and cope with such artifacts. Most of the previous work has concentrated on detecting a specific artifact like rain drops on the windshield of a car. In this paper, we show that on moving systems most artifacts can be detected by analyzing the frames in a stream of images from a camera for static image parts. Based on the observation that most artifacts are temporally stable in their position in the image we compute pixel-wise correlations between images. Since the system is moving the static artifacts will lead to a high correlation value while the pixels showing only scene elemets will have a low correlation value. For testing this novel algorithm, we recorded some outdoor data with the three different artifacts: raindrops, dirt and scratches. The results of our new algorithm on this data show, that it reliably detects all three artifacts. Moreover, we show that our algorithm can be implemented efficiently by means of box-filters which allows it to be used as a self-checking routine running in background on low-power systems such as autonomous field robots or advanced driver assistant systems on a vehicle.**

I. INTRODUCTION

In the last years there was a strong development of robotic systems, in particular autonomous systems that move through their environment without a human operator. Many of these autonomous systems use cameras to perceive the environment because cameras yield very detailed information and due to their passive nature have a low power consumption. In some domains these systems have already matured from pure research into real products like *advanced driver assistant systems* (ADAS) available on most modern cars. However, when a camera's view gets obstructed, e.g. by dirt on the lens, then performance of the system can be deteriorated or the system might fail. Because of this it is important that such systems can run self-checks in order to evaluate if they are still able to fulfill their designated function.

In this paper, we show that it is possible to reliably detect different artifacts in both field and street environments. Our basic assumption is that artifacts are typically temporally static in the images of a stream from a camera mounted on a moving system. By finding the static parts in the stream of images we can detect the image artifacts independent of their actual nature. Of course this only works when the scene is changing, i.e. when the camera is moving. However, for the targeted application of autonomous field robotics and ADAS it is easy to access ego-motion information of the system in order to only process images that were taken during motion.

[1]Nils Einecke and Jörg Deigmöller are with the Honda Research Institute Europe GmbH, D-63073 Offenbach/Main, Germany nils.einecke@honda-ri.de, joerg.deigmoeller@honda-ri.de

[2]Harsh Gandhi is with the Darmstadt University of Applied Sciences, D-64295 Darmstadt, Germany harsh598@gmail.com

II. RELATED WORK

In the past, most work focused on detecting rain drops on the lens or a windshield. In [1] rain drops are detected by comparing rendered rain drops with actually perceived ones. This is based on a very elaborated photometric raindrop model. The major drawback of this approach is its high computational complexity and that the raindrop model assumes a specific raindrop shape. Raindrops differing from that shape are not detected. The constraints on the raindrop shape were relaxed in [2] by using Bézier curves instead of a circular model but the computational effort is still very high. Another approach for raindrop detection is presented by Kurihata et al. [3]. Here a raindrop model is learned by means of PCA analysis which is then used to detect raindrops. Unfortunately, it is a large effort to generate the training images and the results show that the approach struggles with textured backgrounds.

On the other hand, some approaches also use intensity variations for detecting rain drops. In [4] the onset of raindrops is detected by analyzing the stream of images of a camera. The onset of a raindrop is detected when there is a strong intensity change from frame $n-1$ to the frame n but only a small intensity changes from frame n to $n+1$. The idea is that raindrops suddenly appear and roughly stay at the same spot in the images. Similarly You et al. [5] analyze the changes between images from an image stream. If there are areas with little changes in intensity these are likely to be raindrops. The underlying idea is that raindrops act as an additional lens thereby compressing image information which leads to slower intensity changes over time as compared to other image areas. In contrast in [6] raindrops are detected by first generating an intensity gradient map which is then thresholded into a binary map. Secondly, the binary map is segmented into connected components which are then analyzed for certain shape properties. The major drawback of analyzing intensity variations is that these are prone to artifacts when the lighting conditions change. This may happen due to changes in the camera like exposure or gain or due to environmental changes like moving from a sunny region into a shadow region.

There is also work for detecting other artifacts. In [7] sensor dust is detected by correlating the camera images with a set of Gaussians with different small σ's. In order to reduce the false positive rate the detected dust spots are

further analyzed. All spots that do not have a favorable round shape, that do not monotonically decrease around the center or that are surrounded by multiple minima are removed from the detection map. The detection of dust spots works reliably; however, the approach itself does not allow for an easy extension for detecting other artifacts then sensor dust. For detecting dirt on a camera lens Chen and Fuh [8] first generate an ideal vignette image for a certain camera. Then they use this ideal vignette image to subtract it from an acquired vignette image. At pixels polluted with dirt there will be a high difference between the ideal and acquired vignette image. Although the approach might generalize to other artifacts it can hardly be used in real-world applications as acquiring a vignette image requires a special test setup. In [9] environmental fog is recognized by detecting the backscattered veil that is induced by the ego-lights of a car and by detecting halos around external light sources. The backscatter detection is done by correlating camera images to a pre-taken reference image. The whole detection consists of a cascade of steps including intensity thresholding, segmentation and some shape checks. Unfortunately, this approach is very much tailored to detecting fog in street environments and thus neither generalizes to other environments nor to other artifacts. Garg and Nayar [10] presented an approach for detecting and removing falling rain. Their basic idea is that falling raindrops generate a specific photometric dynamic which can be detected by means of a temporal correlation. The downside is that the temporal rain pattern can only be detected if the background is stationary which limits the whole approach mainly to stationary cameras.

Work on the general detection of artifacts is quite scarce. In an approach presented by Alippi et al. [11] artifacts in cameras are detected by analyzing the level of blur and by employing a statistical change detection for detecting strong changes in blur. The major drawback of this technique is that it assumes blur to be an artifact. This, however, only holds true for static cameras. In ADAS and field robotics the cameras are moving and thus might induce motion blur, especially at night when the exposure rates of the cameras have to be quite high. Furthermore, the blur change detection requires a set of reference frames without blur from a scene that is statistically similar to the observed scene which is also quite difficult to guarantee for moving cameras.

III. ARTIFACT DETECTION

As described in the last section most attempts for detecting artifacts in camera images are concentrated on detecting raindrops on windscreens. Furthermore, approaches for other artifacts are often tailored very closely to a certain application. Hence, there is almost no work on general artifact detection. However, the approach presented by You et al. [5] is very interesting because of the assumption it is based on. You et al. argue that areas obscured by raindrops change their intensity much slower over time as compared to non-obscured areas. The reason is that a raindrop acts as an additional lens which creates a compact image of the surrounding with a scaling factor of 20 to 30. Due to this also the motion observed in the raindrop image is reduced by factor 20 to 30. In addition, if the raindrop is out of focus the image seen through the raindrop will be blurred which further decreases the intensity changes over time. From these observation You et al. derive a simple algorithm for raindrop detection which uses temporal intensity differences.

In this paper, we argue that a similar approach can be used for a general artifact detection. Our idea is based on the observation that the image position of most artifacts is stable at least for a certain period of time. Thus, artifacts can be detected by analyzing consecutive images for static image parts. Of course this will only work when the camera is moving but typically inertia sensors or other movement sensors are available on field robots or vehicles so that images taking during a stand still can be neglected.

A problem that still remains is that the apparent movement of scene elements in the images depends on their distance to the camera. The reason is that pixels that correspond to real world coordinates at far distance undergo (almost) no motion on the image plane. From the pinhole camera model we know that the projection $\mathrm{x} = (x, y)^T$ of a real world coordinate $\mathrm{X} = (X, Y, Z)^T$ on the image plane is

$$\mathrm{x} = \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{X}{Z} \\ \frac{Y}{Z} \end{pmatrix} . \tag{1}$$

Let's now consider that the camera is moving straight forward for ΔZ meters. This means that for all scene elements the distance to the camera changes also ΔZ which leads to an apparent motion $\mathrm{v} = (u, v)^T$ on the image plane. The apparent motion is connected to the real world motion by

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{X}{Z-\Delta Z} \\ \frac{Y}{Z-\Delta Z} \end{pmatrix} - \begin{pmatrix} \frac{X}{Z} \\ \frac{Y}{Z} \end{pmatrix} = \begin{pmatrix} \frac{x}{\frac{Z}{\Delta Z}-1} \\ \frac{y}{\frac{Z}{\Delta Z}-1} \end{pmatrix} \tag{2}$$

From the apparent motion equation 2 we can see that if $\frac{Z}{\Delta Z} >> x$ and $\frac{Z}{\Delta Z} >> y$, the motion vector v becomes zero. This obviously happens at far distant points. For example trees on an open field can be hundreds of meters away which easily causes the apparent movement to become less than a pixel and hence almost unnoticeable. Because of this we propose to use images taken during a turning or curve maneuver.

If the camera undergoes a pure rotational motion around the focal point then all scene elements have an apparent movement $\mathrm{v} = (u, v)^T$ on the image plane that is independent of their distance to the camera. For now let's consider a rotation of α around the Y-axis because this is the most common camera rotation for ground-based platforms. The apparent movement v on the image plane then is:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{\cos(\alpha)X+\sin(\alpha)Z}{-\sin(\alpha)X+\cos(\alpha)Z} \\ \frac{Y}{-\sin(\alpha)X+\cos(\alpha)Z} \end{pmatrix} - \begin{pmatrix} \frac{X}{Z} \\ \frac{Y}{Z} \end{pmatrix} . \tag{3}$$

If we divide the first term in this equation by Z we get

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{\cos(\alpha)\frac{X}{Z}+\sin(\alpha)}{-\sin(\alpha)\frac{X}{Z}+\cos(\alpha)} \\ \frac{\frac{Y}{Z}}{-\sin(\alpha)\frac{X}{Z}+\cos(\alpha)} \end{pmatrix} - \begin{pmatrix} \frac{X}{Z} \\ \frac{Y}{Z} \end{pmatrix} . \tag{4}$$

Using the projection equation 1 we can replace all world coordinates with image coordinates

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{\cos(\alpha)x+\sin(\alpha)}{-\sin(\alpha)x+\cos(\alpha)} \\ \frac{y}{-\sin(\alpha)y+\cos(\alpha)} \end{pmatrix} - \begin{pmatrix} x \\ y \end{pmatrix} . \quad (5)$$

We can see from the apparent motion equation 5 that the motion vector v is independent of the depth Z and can only get zero if the camera is not rotating ($\alpha = 0$). For a rotation about the X-axis the derivation can be done in an analogous way. For the rotation about the Z-axis the derivation is a bit different but it also is obvious that the apparent movement in this case is not dependent on depth. Please note that the above derivation holds only true for rotations about the focal point. This, however, is rarely the case. Fortunately, the influence of depth can be neglected for most turning maneuvers (turning at the spot) and is also quite weak for curve driving.

It is important to highlight that the theoretical thoughts above not only holds for pure rotational motion. A rotational motion component always causes a depth-independent pixel motion, even in combination with translational motion.

In summary, we propose to detect image artifacts by searching for static image elements when there actually should be none. This is only guaranteed when the camera is moving and from a theoretical point of view is done best if the camera undergoes a rotational motion or a combination of translational and rotational motion.

For the actual detection one could think of using also temporal intensity differences as proposed in [5] for raindrop detection but intensity differences are very sensitive to changing lighting conditions which occur quite often in the real world. For example when the sun appears from behind a cloud suddenly the overall luminance in the scene will increase, leading to strong differences in intensity even for consecutive frames. Because of this, we propose to measure the change in structure instead of the change in intensity.

In this work, we use the *normalized cross-correlation* (NCC) because it has proven to be very robust against illumination changes [12] and is cheap to compute [13]. This means that, instead of simply computing intensity differences

$$\Delta I_{1/2} = |I_1(x) - I_2(x)| , \quad (6)$$

we calculate pixel-wise correlation coefficients in a block-matching style

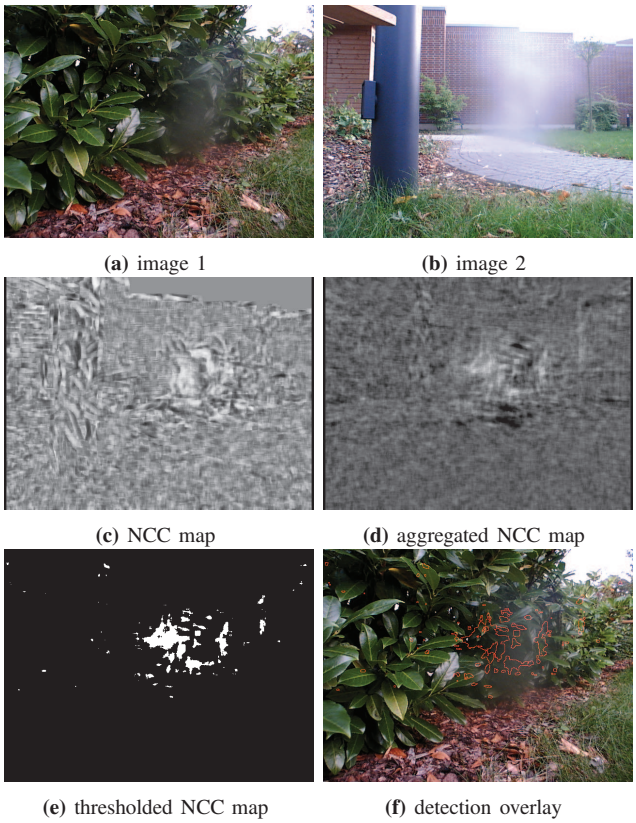
$$\rho_x = \frac{1}{|p(x)|} \sum_{x' \epsilon p(x)} \frac{(I_1(x') - \mu_1(x))\,(I_2(x') - \mu_2(x))}{\sigma_1(x)\sigma_2(x)} , \quad (7)$$

where

$$\mu_j = \frac{1}{|p(x)|} \sum_{x' \in p(x)} I_j(x') , \quad (8)$$

$$\sigma_j = \sqrt{\frac{1}{|p(x)|} \sum_{x' \in p(x)} (I_j(x') - \mu_j)^2} . \quad (9)$$

Please note that other measures insensitive to lighting like variants of NCC [13] or rank transformed images or census



(a) image 1 **(b)** image 2 **(c)** NCC map **(d)** aggregated NCC map **(e)** thresholded NCC map **(f)** detection overlay

Fig. 1: This figure shows an example of the artifact detection for a camera having scratches near the center of the lens. The images in (a) and (b) are two frames taken during a turning maneuver. These two images are 100 frames apart. In (c) the NCC map for a pixel-wise correlation of (a) and (b) with an 11x11 match-window is shown. Bright values are high correlations and dark values are low correlations. The artifact is clearly visible in this correlation but there is also a lot of measurement noise. Thus it is more favorable to accumulate some NCC maps over time. In (d) 100 NCC maps have been accumulated. Here the artifact area is much more pronounced with respect to the rest of the image and allows for a robust detection. For actually detecting an artifact the accumulated NCC map is thresholded as shown in (e) and the binary map is summed up. If the number of static pixels exceeds a second threshold then an artifact detection is signaled. For visualization (f) shows an overlay of image (a) and the binary detection map (e).

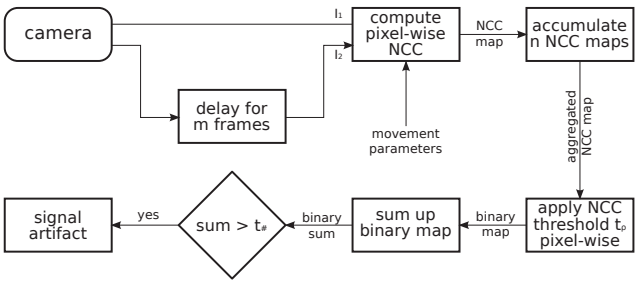
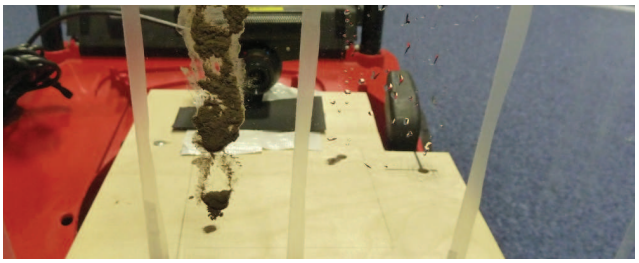


Fig. 2: Algorithm overview.

(a) multi-camera

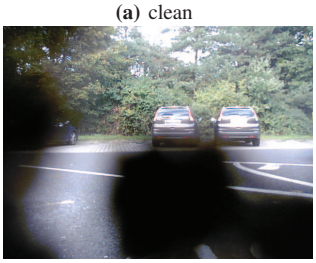


(b) glass pane

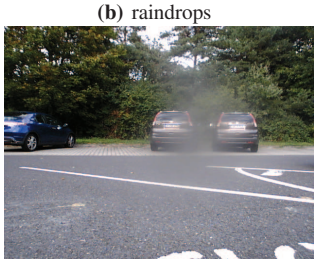
Fig. 3: This figure shows the two setups we used in our experiments. On the one hand we used (a) a multi-camera setup consisting of four PSEye cameras each having a different artifact and one having a clear view. On the other hand (b) we used a single camera behind a glass pane which had three sections polluted with the different artifacts and one section with a clear view.

(a) clean

(b) raindrops



(c) dirt



(d) scratches

Fig. 4: This figure shows the view from the four cameras we used in our experiments. While (a) one camera has a clean view, the other three cameras have different artifacts (b) raindrops, (c) dirt and (d) scratches. Please note that the raindrops on the lens are hard to see as they mainly manifest in blurred artifacts, seen in (b) at the middle right and bottom left.

Fig. 5: This image is taken from the camera in the glass panel setup. On the panel we marked four areas with tape. From left to right the first area has scratches, the second has raindrops, the third has dirt and the last one has a clear view.

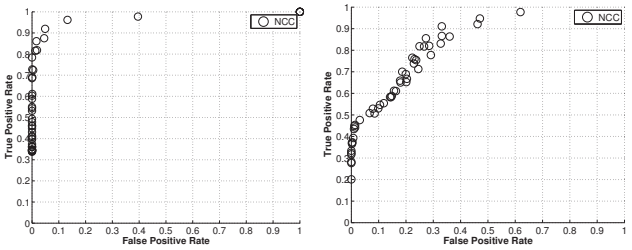
transformed images [14] are also applicable here. Furthermore, please note that we convert color images to gray images before computing the correlations because it has been shown that color does not improve the correlation performance [15].

Fig. 1 shows a small example of a camera with a scratch artifact near the center of the lens. The two images shown in this figure are taken 100 frames apart during a turning maneuver to the right. Both images show a blurring-like artifact due to the scratches on the lens. Applying the pixel-wise correlation we propose leads to the correlation map shown in Fig. 1c. In this map the artifact of the camera is clearly visible by the high correlation values (bright pixels) in the correlation map. Unfortunately, by chance some structure in two images can be very similar leading to the strong correlation noise in Fig. 1c. Because of this we propose to accumulate the NCC map over time in order to pronounce the artifacts and to reduce the correlation noise. The result of 100 NCC maps accumulated can be seen in Fig. 1d. In this accumulated NCC map the artifact is still clearly visible while the correlation noise in the rest of the image is suppressed.

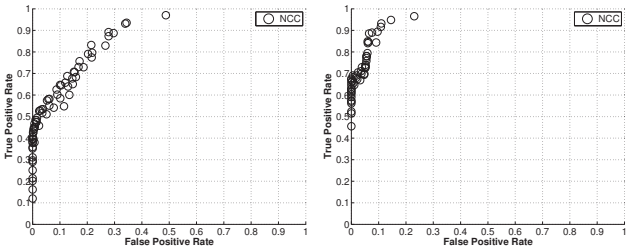
For actually detecting an artifact we first threshold the accumulated NCC map with a certain threshold t_ρ. In the resulting binary map (see Fig. 1e) we then count the number of active pixels. If this count exceeds a second threshold $t_\#$ then we regard the current image frame to show an artifact. To summarize our approach Fig. 2 gives on overview of the single processing steps.

IV. EXPERIMENTAL RESULTS

For testing our new approach we recorded roughly 18500 images in two different setups and for two different environments. On the one hand, we used a multi-camera (see Fig. 3a) setup consisting of four PSEye cameras where three of the cameras had three different artifacts (raindrops, dirt, scratches) and one camera had a clear view. The recordings from these cameras was synchronized, i.e. we acquired temporally synchronized images so that besides some small variations in viewing point the image from different cameras with the same frame number show the same scene. To get a better idea of the resulting image quality Fig. 4 shows a sample of an image set from the synchronized cameras.

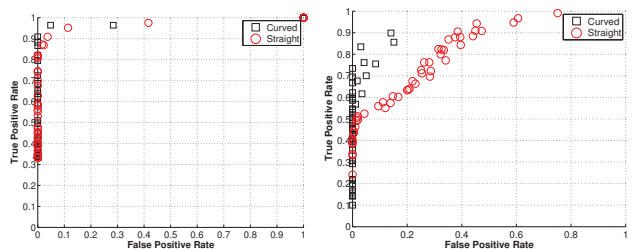


(a) multi-camera on field **(b)** multi-camera on street

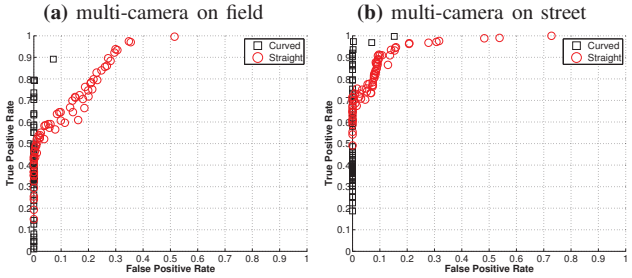


(c) glass panel on field **(d)** glass panel on street

Fig. 6: These plots show the artifact detection performance for the two scenarios *field* and *street* and the two setups *multi-camera* and *glass panel*. The best performance has been achieved in multi-camera setup on the field. Using the glass panel setup the detection was sometimes distracted by glares. This problem was less prominent in the car on the street as the cars hull reduced the stray lights from the side. However, in the street scenario the detection struggles with visible clouds that hardly move during straight drives. This is most prominent in the multi-camera setup because here the field of view is the full camera view while in the glass panel setup it is restricted to a smaller area.



(a) multi-camera on field **(b)** multi-camera on street



(c) glass panel on field **(d)** glass panel on street

Fig. 7: These plots show the difference in the accuracy of artifact detection for frames taken during straight drive and frame taken during turning or curve maneuvers. For the multi-camera setup in the field scenario (a) the difference is less prominent as the overall performance was already quite high. In the street scenario (b) and (d) where lot of clouds are visible in the images the separation increases the performance tremendously. Moreover, the separation also increases the performance for the glass panel setup in the field which struggled with glares before. This demonstrates that artifact detection is best done when the camera undergoes a rotational movement.

Although the cameras are capable of 60Hz in VGA resolution we had to reduce the capturing frame rate to 15Hz as otherwise the limits of the USB bus would have hampered a synchronous capturing of all four cameras. On the other hand, we used a single PSEye camera (see Fig. 3b) behind a glass panel. The panel was divided into four vertical sections where three sections had three different artifacts (raindrops, dirt, scratches) and one section had a clear view. Fig. 5 shows a sample view of the camera through the glass panel.

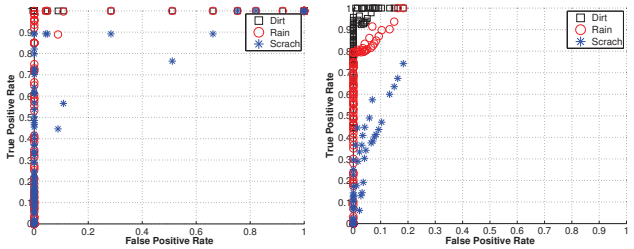
In order to capture a larger variety of scenes we took images in two different environments. The first environment is a grass area which is meant to simulate environments that typical field robots would encounter in the open world. The second environment is an urban street scene taken from a driving car in order to test the performance for ADAS. In both environments we did recordings with both the multi-camera and the glass panel setup.

Since our target is to have an algorithm performing a self-check in a camera-based system we evaluate our novel algorithm by assessing its performance for detecting the presence of artifacts in the camera images. In the multi-camera setup the ground truth is directly given by the camera ID (each having either a certain artifact or a clear view). For evaluating the glass panel images we labeled the four sections by manually generating binary masks for each section. Please note that we concentrate on the detection of artifacts here. This means we do not discuss correction schemes. Rather we think that using our algorithm we could trigger appropriate countermeasure actions which in ADAS could be as simple as informing the driver (e.g. in case of a rear-view camera) or automatically starting the wiper (e.g. in case of a front looking camera behind the windshield), or in field robotics trigger a camera cleaning action or another algorithm trying to correct the artifacts or marking certain image parts as unreliable.

In all the following experiments, we use the receiver operating characteristic (ROC) to evaluate our algorithm. The ROC plots are generated by varying the two free parameters t_ρ and $t_\#$, i.e. the threshold for the NCC map binarization and the pixel number threshold for the final artifact detection. For all these experiments we varied t_ρ from 0.2 to 0.5 with a step size of 0.05. The threshold variation for $t_\#$ was 1000 to 10000 with a step size of 1000 for the glass panel setup and 8000 to 32000 with a step size of 4000 in the multi-camera setup. Note that we had to use a smaller pixel number threshold in the glass panel setup due to the separation of the glass panel (and hence the camera image) into the four segments.

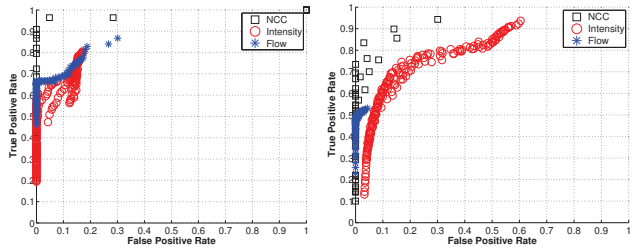
A. General Performance

In a first experiment we tested the general performance of the system. Fig. 6 shows the ROC curves for the two environments *field* and *street*, and the two setups *multi-camera* and *glass panel*. The first thing that strikes is that the performance for the multi-camera setup in the field is best. This means that lens artifacts are reliably detected for field-like scenarios. In contrast the artifact detection on the glass panel shows significantly lower performance. The major reason for this are some glares caused by the sun.



(a) artifact comparison on field **(b)** artifact comparison on street

Fig. 8: This figure shows the ROC curves for the different artifacts raindrop, dirt and scratch. In both (a) the field and the (b) the street scenario, dirt and raindrops are reliably detected. In contrast, the scratches are harder to detect. While in the field some settings still allow for a good 0.9 TPR at zero FPR, in the street scenario the TPR for zero FPR does not exceed 0.4 TPR. Please note that these results are for frames from curved maneuvers only.



(a) algorithm comparison on field **(b)** algorithm comparison on street

Fig. 9: These ROC curves show a comparison for three different algorithms for static image artifact detection: our correlation-based algorithm (NCC), a simple intensity difference (Intensity) and optical flow (Flow). The results disclose that the artifact detection based on cross-correlation values works much better than intensity difference or optical flow.

In the street scenario the glares were less prominent in the data. This is because we took the recordings from the trunk of a car so that the car's hull reduced stray lighting from the side. Hence, the performance for detecting artifacts on a protection glass is much better in the street scenario than in the field scenario. On the other hand in the street data a lot more clouds are visible due to the higher viewing directions of the cameras as compared to the field recordings. This leads to false artifact detections while driving straight because the clouds hardly move. We already discussed this problem in section III. From the pinhole camera model follows that scene elements with a large distance will have no significant apparent movement in the images over a short time period (see also the apparent movement equation 2 in section III). Thus we already argued to concentrate the artifact detection to image frames that were taken during turning or curve maneuvers.

In order to test how the camera dynamics influence the performance we manually separated our test recordings into frames taken during straight drive and frames taken during turnings or curved drive. The results of this analysis are depicted in Fig. 7. In the field scenario with the multi-camera setup there is only a slight improvement for the curved frames as the performance was already very high for the overall stream. However, in the street scenario (both with the multi-camera and the glass panel setup) there is a significant difference between the frames recorded during straight drive and frames recorded during turning drives. As discussed above the major reason are the clouds visible in the street data. Since clouds are far away they will show only very little motion in the image when the system is driving straight thus they easily lead to false positives. In contrast during turning or curve drive the distance to an object has little or no effect on the apparent pixel motion (see the apparent motion equation 5 in section III). The analysis also revealed that the detection performance is strongly increased for the glass panel setup on the field where the system struggled with glares beforehand. We think the reason is that the position of a glare is dependent on the direction of the sun. During straight drive the direction of the sun is almost constant while during turning it changes which causes the glare to move during turn maneuvers. In summary, the results support our argumentation that the artifact detection is best done on frames taken during turning or curve maneuvers.

B. Dirt, Rain and Scratch Detection Performance

In the second experiment we assess how good the detection performance is for the different artifacts. Fig. 8 plots the performance for raindrops, dirt and scratches. In the field scenario the performance for the three types of artifacts is very high; however, in the street scenario the algorithm struggels to detect the scratch artifacts. This seems a bit disappointing at first but when we looked closer at the image streams we realized that the scratch artifact is much less pronounced in the street image data than it is in the field data. As discussed above, this is because the cameras are better shielded from stray light in the car than they are in the open field. Due to the reduced stray light in the car the scratches on the camera lens also diffract less light which reduces the artifacts in the images. This also means that in this case it is not bad that the artifact detection is stuggeling to detect the scratch artifact since the impact on the image quality is low anyway.

C. Comparison to other approaches

The third experiment compares the performance of the proposed NCC measure to other approaches. Due to a lack of methods for general artifact detection for moving systems we compare our method to two other methods for detecting static image elements. The first is a standard difference image approach. This corresponds to the raindrop detection in [5] without the level-set segmentation step. Unfortunately, we realized that this approach works bad in real world scenes because of overexposed image areas. Although the overexposed areas also move over time they are still likely to overlap after some frames. This causes false positive detection of artifacts because the intensity difference is zero. To overcome this problem, we ignore an artifact detection at a pixel position (x, y) in case both pixels in $I_1(x, y)$ and $I_2(x, y)$ are above an intensity value of 250. As second method, we compute an optical flow with a small search range of ± 2 shifts. We restricted the flow to this small

search range because we concentrate on methods that can be computed on platforms with low computational power. The optical flow allows to evaluate if a patch is really static or if there is a patch in the close neighborhood with higher correlation. For computing the optical flow we also use the normalized cross-correlation measure. However, artifacts are not detected based on the correlation value but based on an optical flow of zero. Due to the small search range we compare neighboring frames instead of images that are 100 frames apart, as we do in the NCC and intensity difference algorithm.

The result of the comparison of the three different algorithms is shown in Fig. 9. The ROC plots show that our static image element detection based on normalized cross-correlation work much more reliable than the intensity difference or optical flow. The worse performance of the intensity difference was expected because these are very prone to fail in real world lighting conditions. On the other hand, the strongly worse performance of the optical flow is a bit surprising since we also used cross-correlation to compute the flow. However, our findings are similar to those in [5] where SIFT-based optical flow turned out to be inferior to intensity differences for raindrop detection. We have no completely satisfying explanation for the weaker performance of optical flow but we think that it is caused by the small search range and the comparison of neighboring frames with a small Δt in our case.

D. Computational Complexity

One important thing about detecting artifacts in camera images is that it should be efficient so that it can run as a self-checking routine in the background. The most time consuming part of our algorithm is the pixel-wise correlation because it requires to compute the mean and variance for each pixel. For the sake of efficiency we compute these values using quadratic regions around each pixel. Thus the pixel-wise mean and variance values can be computed with box-filters which can be computed efficiently using running sum and separated filtering [16]. This requires only two subtraction and two addition operation per pixel, independent of the used box-filter size. Please note that we do not use integral images [17] (summed area tables [18]) to compute the box-filters because they require more operations per pixel and they are numerically less stable because of the full summation of all image values.

We implemented our algorithm in plain C and measured the runtime on three different systems. First, we used a standard desktop machine with with a 3GHz CPU. Second, we used two different embedded boards (Odroid-X2 and Arndale) which are more suited for autonomous systems because of the lower energy consumption. Table I shows a small overview of the computation time for correlating two images. These times were measured using only one core for each tested processor. The pure runtime for VGA images seems to be quite high on the embedded boards. However, we also need to consider that the correlation does not need to be computed for every image. Furthermore, there is a lot room for improvement for example by using the NEON instruction set available on modern ARM CPUs or by sharing the mean and variance values with other algorithms like stereo or optical flow.

TABLE I: This table gives a small overview of the computation time for correlating two images. All runtimes were measured for a plain C code implementation running on a single core.

	CPU 3GHz	odroid	arndale
640x480	23ms	90ms	98ms
320x240	5ms	21ms	25ms

V. CONCLUSION

In this paper, we presented a general approach for detecting artifacts like dirt, raindrops or scratches in the camera images of outdoor mobile (moving) systems like robots or cars. The idea is to provide such mobile systems with computationally cheap yet robust means for performing a self-check of their vision. We start from the basic assumption that usually artifacts are stable in their image position at least for a short period of time. Thus the artifacts will not move even if the whole observed scene changes. By pixel-wise correlating images from different time steps with the normalized cross-correlation (NCC), static image elements can be extracted. The easiest way to realize that the scene is changing is to analyze images that were acquired during the motion of the camera. Information on whether the whole system is moving should be available on most modern robots or cars.

For testing our algorithm, we recorded several minutes of data for two different setups in two different environments. First, we employed a synchronized multi-camera setup to test the detection of artifacts in camera lenses. Second, we used a camera behind a glass panel to test the detection of artifacts on a camera protection glass. With both setups we recorded data on a grass field and in an urban street environment. Our experiments showed that our algorithm is able to reliably detect artifacts in both setups and both environments. The experiments also supported our theoretical discussion that artifacts are best detected during turning or curve maneuvers because only then one can guarantee that the whole scene is changing while only the artifacts do not move. We also compared our approach of correlating images with NCC to an intensity difference approach and an approach based on optical flow. Both intensity difference and optical flow showed a performance inferior to our correlation approach. Furthermore, we showed that our algorithm can be implemented very efficiently, taking only a few milliseconds to check one image pair.

Although, we did a thorough investigation of our approach for raindrop, dirt and scratch detection there are still some things that need to be evaluated in the future. First, we need to do some more test on scratch detection as our experiments showed some weakness of our algorithm in detecting scratches especially in the street scenario. Second, we also would like to test our algorithm for other camera artifacts that might hamper vision algorithms like lens-fog

or small objects like leaves that might stick on the lens. Last but not least an important step is to integrate our algorithm into an existing field robot or vehicle system in order to test our algorithm under more natural conditions and to get a larger data set. This will also help us to understand how to communicate the detected artifacts to other processing modules which is important for a successful application in different systems.

REFERENCES

[1] M. Roser and A. Geiger, "Video-based raindrop detection for improved image registration," in *International Conference on Computer Vision (ICCV) Workshops*, 2009, pp. 570–577.

[2] M. Roser, J. Kurz, and A. Geiger, "Realistic modeling of water droplets for monocular adherent raindrop recognition using bézier curves," in *ACCV Workshops*, 2010, pp. 235–244.

[3] H. Kurihata, T. Takahashi, I. Ide, Y. Mekada, H. Murase, Y. Tamatsu, and T. Miyahara, "Rainy weather recognition from in-vehicle camera images for driver assistance," in *Intelligent Vehicles Symposium*, 2005, pp. 205–210.

[4] F. Nashashibi, R. de Charette, and A. Lia, "Detection of unfocused raindrops on a windscreen using low level image processing," in *11th International Conference on Control Automation Robotics & Vision (ICARCV)*, 2010, pp. 1410–1415.

[5] S. You, R. Tan, R. Kawakami, and K. Ikeuchi, "Adherent raindrop detection and removal in video," in *26th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 1035–1042.

[6] A. Cord and D. Aubert, "Towards rain detection through use of in-vehicle multipurpose cameras," in *Intelligent Vehicles Symposium*, 2011, pp. 833–838.

[7] A. Dirik, H. Sencar, and N. Nasir, "Digital single lens reflex camera identification from traces of sensor dust," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 3, pp. 539–552, 2008.

[8] C.-W. Chen and C.-S. Fuh, *Pattern Recognition, Machine Intelligence and Biometrics*. Springer, 2011, ch. 7 Lens Shading Correction for Dirt Detection, pp. 171–198.

[9] R. Gallen, A. Cord, N. Hautière, and D. Aubert, "Towards night fog detection through use of in-vehicle multipurpose cameras," in *Intelligent Vehicles Symposium*, 2011, pp. 399–404.

[10] K. Garg and S. Nayar, "Detection and Removal of Rain from Videos," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. I, 2004, pp. 528–535.

[11] C. Alippi, G. Boracchi, R. Camplani, and M. Roveri, "Detecting external disturbances on the camera lens in wireless multimedia sensor networks," *IEEE T. Instrumentation and Measurement*, vol. 59, no. 11, pp. 2982–2990, 2010.

[12] H. Hirschmüller and D. Scharstein, "Evaluation of stereo matching costs on images with radiometric differences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 9, pp. 1582–1599, 2009.

[13] N. Einecke and J. Eggert, "A two-stage correlation method for stereoscopic depth estimation," in *International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, 2010, pp. 227–234.

[14] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence," in *Proceedings of Third European Conference on Computer Vision (ECCV)*, 1994, pp. 151–158.

[15] M. Bleyer and S. Chambon, "Does color really help in dense stereo matching?" in *International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, 2010, pp. 1–8.

[16] M. J. McDonnell, "Box-filtering techniques," *Computer Graphics and Image Processing*, vol. 17, no. 1, pp. 65–70, September 1981.

[17] P. A. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[18] F. Crow, "Summed-area tables for texture mapping," in *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, 1984, pp. 207–212.