Doing More With Moiré Pattern Detection in Digital Photos

Citation for published version:
https://doi.org/10.1109/TIP.2022.3232232

Digital Object Identifier (DOI):
10.1109/TIP.2022.3232232

Link:
Link to publication record in Heriot-Watt Research Portal

Document Version:
Publisher's PDF, also known as Version of record

Published In:
IEEE Transactions on Image Processing

General rights
Copyright for the publications made accessible via Heriot-Watt Research Portal is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
Heriot-Watt University has made every reasonable effort to ensure that the content in Heriot-Watt Research Portal complies with UK legislation. If you believe that the public display of this file breaches copyright please contact open.access@hw.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
Doing More With Moiré Pattern Detection in Digital Photos

Cong Yang, Zhenyu Yang, Yan Ke, Tao Chen, Marcin Grzegorzek, and John See

Abstract—Detecting moiré patterns in digital photographs is meaningful as it provides priors towards image quality evaluation and demoiréing tasks. In this paper, we present a simple yet efficient framework to extract moiré edge maps from images with moiré patterns. The framework includes a strategy for training triplet (natural image, moiré layer, and their synthetic mixture) generation, and a Moiré Pattern Detection Neural Network (MoireDet) for moiré edge map estimation. This strategy ensures consistent pixel-level alignments during training, accommodating characteristics of a diverse set of camera-captured screen images and real-world moiré patterns from natural images. The design of three encoders in MoireDet exploits both high-level contextual and low-level structural features of various moiré patterns. Through comprehensive experiments, we demonstrate the advantages of MoireDet: better identification precision of moiré images on two datasets, and a marked improvement over state-of-the-art demoiréing methods.

Index Terms—Moiré pattern, moire pattern detection, moire removal, moire image restoration, adaptive kernel.

I. INTRODUCTION

Moiré effects seen in digital photography are the result of spatial interference between the pixel grids of the camera sensor and the device screen [1] – a long-standing problem in computer vision. While existing methods mainly concentrate on demoiréing (also called moiré effect minimization, moiré removal, moiré photo restoration, and moiré mopping in some literature) [2], [3], [4], [5], [6], few studies have been devoted to moiré pattern detection. Moiré pattern detection and demoiréing are two different tasks: demoiréing task seeks to remove moiré patterns to reveal an underlying clean image, while moiré pattern detection aims to detect the shape, location, and intensity of moiré patterns within an image.

In practice, moiré pattern detection is required in a variety of real-world applications. For instance: (1) Face-spoofing detection. A smart-lock should have the ability to determine whether the unlocking face is real, or just displayed from screens (see Fig. 20). In such a case, the detected moiré patterns are extremely meaningful for the determination [7], [8], [9], [10]. (2) Recapture detection. The detected moiré patterns can be used for recapture detection in smart retail scenarios. This is motivated by the fact that some floorwalkers try to cheat the SKU (Stock Keeping Unit) recognition system via phone-captured screen images, rather than real images captured on-site. (3) Guideboard analysis for autonomous driving. In practice, distinguishing digital and physical guideboards is meaningful, as it correlates to the strategies used for guideboard context analysis. It is also necessary to eliminate advertising boards that might be misleading. In such a case, the detected moiré patterns on digital boards can be used for determination. (4) Fashion design. Some textiles (e.g. the leftmost image in Fig. 15) could have distinct moiré effects with particular camera poses. The detection results provide valid information to highlight (or avoid) such effects during the fashion show drill. (5) Media production. For example, automatically detecting the screen area (aka. the area with dense moiré patterns) and proceeds to cover it via mosaicing (or blurring) for privacy purposes.

Moiré pattern detection can also be used to improve the usability and performance of demoiréing. Particularly, existing demoiréing methods are applied to the whole image, while in practice the area that needs demoiréing may only be part of the image (e.g. the leftmost image in Fig. 1). In such a case, demoiréing the screen area is actually “blurring”, rather than “denoising”. The detected moiré patterns provide prior information on the moiré pattern distribution and density. Such information is valuable; it can be further used to estimate the intensity and localize the area that requires demoiréing.

Inspired by image reflection separation [11] and skeleton extraction [12], [13], moiré pattern detection could be regarded as a translation task from moiré image to moiré patterns. For instance, similar to reflection, moiré patterns are fused with the context and texture of their backgrounds. In such a case, it is...
difficult to directly separate moiré patterns and the background using traditional filtering methods. Inspired by the supervised methods of image reflection separation, deep neural networks are proposed for such translation tasks. This is motivated by the fact that both reflection, background, and their mixture are provided for training the neural network. Accordingly, a training triplet consisting of a natural background image, moiré layer and their synthetic mixture, can be generated for the purpose of training a model for various related tasks.

However, it is a challenging problem to generate training triplets and detect moiré patterns using existing approaches. Particularly, there is still no accurate mathematical modules to describe moiré patterns, especially in camera-captured screen images. This is because moiré patterns are extremely sensitive to the variations in camera poses, screens, scales and lights [14]. As a result, intricate patterns are commonly spread across various frequency bands of images with varying colors, shapes and intensities. Thus, it is challenging to extract moiré layers for synthetic mixture generation. Though some hand-crafted features have been introduced to model simple moiré patterns [15], [16], [17], those methods are proposed in specific domains and have limited capabilities at modelling complex patterns found in camera-captured screen images. Recently, mosaic-based simulation [3] and corner-based registration [2] approaches were proposed to generate moiré images that were then extended to a simple subtraction approach [5]. However, moiré patterns in the generated layers are either too faint or contain a great deal of noise since these approaches tend to produce distorted samples that are not well-aligned with their original images. In terms of detected patterns, it is somewhat similar to translucent pattern detection scenarios like smoke [18], motion blur [19], shadow [20], haze [21], specular highlights [22] and raindrops [23], etc. However, these approaches cannot be directly employed because unlike the aforementioned scenarios, moiré effects are highly variable and dependent on the intensity and shape of fine-grained moiré patterns. Thus, the appearance of moiré patterns differs greatly in varying scales [1] (see Fig. 2).

To make the problem tractable, we introduce a simple yet efficient framework to generate training triplets and detect moiré patterns. First, we propose a strategy to collect moiré layers from a wide variety of real-world moiré patterns that are further integrated with natural images to generate synthetic moiré images. Our integration method also considers both moiré effects and characteristics of camera-captured screen images. Further, we generate pixel-aligned training triplets by varying natural images, brightness, as well as transparency and perspective transformation of moiré layers. For detection, we propose a Moiré Pattern Detection Neural Network (MoireDet) to exploit both high-level contextual features (i.e. moiré effects) and low-level structural features (i.e. fine-grained stripes, ripples and curves) of various patterns. Significantly, MoireDet utilizes three encoders to properly encode moiré patterns with adaptive kernels that are sample- and location-specific. The efficiency of our framework is validated through two tasks: moiré pattern detection and moiré image identification. We find that our detection results can effectively improve the accuracy of existing moiré image restoration approaches [5], [24], [25].

In summary, our contributions to moiré pattern detection covers two broad aspects. First, we introduce a new strategy that flexibly models moiré patterns to build well-aligned training triplets, which are used for the translation network MoireDet to detect moiré patterns. These two components establish a novel framework that addresses the moiré pattern detection problem. Second, since previous evaluation datasets barely cover moiré pattern detection and identification tasks, two new datasets, MoireScape and MoireIDT, are introduced to encourage the community to develop more algorithms on these scenarios.

II. RELATED WORKS

We briefly discuss several existing works that are relevant to moiré layer generation and moiré pattern detection.

A. Moiré Layer Generation

There are two possible ways to generate moiré layers: direct simulation and extraction from moiré images. For the first way, Oster and Saveljev et al. [15], [16], [26], [27], [28] introduced several models to simulate amplitude, period and orientation of moiré patterns mathematically. However, these methods are proposed to describe simple patterns in specific domains like graphene layers and single-walled nanotubes, etc. They have limited ability to simulate complex moiré effects like those found in camera-captured screen images. For the second way, a moiré layer can be extracted using subtraction operation between a moiré image and its background. For this, Liu et al. [3] proposed a strategy to generate moiré images using a set of operations on natural images such as mosaic resampling, random projective transformation, radial distortion, etc. Based on the similar operations in [3], the LCDMoire dataset [29] was synthetically generated for the purpose of moiré image restoration. Though the generated moiré and original images are aligned pixel-wise, their quality and diversity are somewhat limited: (1) Realism and variety:

---

**Fig. 1.** Top: Camera-captured screen images. Bottom: Detected moiré patterns with the proposed framework.

---

**Fig. 2.** Dynamic moiré effects while rescaling the same moiré image.
Moiré patterns in LCDMoiré are generated by a mosaic-based simulation method and they turn out to be quite different from real-world moiré patterns. Particularly, real-world moiré patterns are composed of various stripes, ripples, and curves, while the simulated moiré patterns are just small uniform-looking stripes. As a result, most moiré effects in LCDMoiré look somewhat similar, only slightly different in shape. Moreover, factors (e.g., camera-screen models, distance and relative poses in Fig. 4) that influence moiré shape and intensity are not well-reflected in the operations. (2) Quality: Though LCDMoiré can ensure pixel-level alignment between a moiré and clean image pair, both images are randomly deformed. This is because their method simulates moiré patterns on clean images with a set of operations such as mosaic resampling and random projective transformation, etc. As a result, there is a big difference between the real and synthetic moiré patterns.

In contrast, Sun et al. [2] introduced a capture-alignment strategy to align the natural and moiré images based on their border corners. Though the image pair can be aligned to some good extent, spatial shifts and colour bias still exist between the pair (even after applying global tone mapping), resulting in tremendous noise coming from the background of the moiré layer. To ensure proper alignment, some works directly extract backgrounds using moiré photo restoration methods [5], [30], [31], [32]. In spite of colour shift in the background, these methods did not generalized well to all images and are particularly weak at handling regional patterns. In contrast, our method not only ensures pixel-wise alignment of training triplets, but also proper preservation of the moiré layer without background contamination. Thus, our training triplets have better quality, which is beneficial to moiré detection, identification and removal algorithms.

B. Moiré Pattern Detection

Though moiré effect minimization and moiré photo restoration have been extensively studied in the past few years [2], [3], [4], [5], [6], [14], [29], [30], [33], [34], few works, however, have been dedicated to moiré patten detection. One possible idea is to extend existing detection methods like smoke [35], blur [19], shadow [20] and haze [36] on moiré patterns since they are naturally akin to translucent pattern detection [37]. For instance, Kim et al. [19] introduced a method that detects motion blur, and defocus the blur and no-blur regions using multi-scale reconstruction loss functions on a neural network that is similarly to U-Net [38]. Further on, approaches that emphasize on the need of different scales were formulated: Hu et al. [20] with a direction-aware spatial recurrent neural network (RNN) for shadow detection; Yuan et al. [39] with the use of coarse and fine smoke segmentation masks generated from fully convolutional networks (FCN). An interesting work by Makarau et al. [36] introduced a dark-object subtraction method to detect and calculate a thickness map of inhomogeneous haze in medium- and high-resolution multi-spectral satellite images. However, unlike these tangible targets that can be clearly observed in most cases, the visibility of moiré patterns is labile and highly dependent on their shape and intensity. Thus, we hypothesize that it is more effective to decode fine-grained stripes, ripples and curves that are related to high-level moiré effects rather than directly detecting from regions in the image – an idea which is positively verified in Section V-C.

We note the works of Abraham [40] and Garcia [9], which proposed to detect moiré patterns using neural network and frequency analysis, respectively. Rather than detecting patterns, these methods were actually designed for moiré image identification. Their binary output are used to classify whether an image is contaminated by moiré patterns. Here, our method not only outputs the patterns, but we also show how they can be easily extended for moiré image identification tasks. Experiments in Section V-D show that our method has promising advantages in terms of flexibility and accuracy compared to these works [9], [40].

III. TRAINING TRIPLETS

We first look at the characteristics of real-world moiré patterns. Built on that, we present our methods for moiré pattern collection and training triplet generation. For better understanding, some essential symbols and their descriptions are summarized in Table I.

A. Characteristics

To structurally explore the interference between grids of display elements and camera sensors (see Fig. 3), we introduce a practical setup for moiré image capture. Particularly, a phone camera is placed in front of a screen using a gooseneck holder. To avoid vibrations caused by touching the phone screen, touch interactions are controlled from a laptop that projects the phone camera in real-time. For each image displayed on the screen, a black border is padded for the purpose of perspective calibration. We then twist the gooseneck holder to simulate five camera poses (top, bottom, left, right, front) with respect to the

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Sections</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>Segmented moiré layer</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>Image background</td>
<td>3</td>
</tr>
<tr>
<td>SM</td>
<td>Binary selection of an original moire layer</td>
<td>3</td>
</tr>
<tr>
<td>t</td>
<td>Perspective Transformation Matrix</td>
<td>3</td>
</tr>
<tr>
<td>\lambda</td>
<td>Empirical threshold</td>
<td>3, 5</td>
</tr>
<tr>
<td>M</td>
<td>Original moire layer</td>
<td>3, 4, 5</td>
</tr>
<tr>
<td>D</td>
<td>Moiré edge map (ground truth)</td>
<td>3, 4, 5</td>
</tr>
<tr>
<td>I</td>
<td>Synthetic moire image</td>
<td>3, 4, 5</td>
</tr>
<tr>
<td>D</td>
<td>Training triplet</td>
<td>3, 4, 5</td>
</tr>
<tr>
<td>G</td>
<td>The sliding window within D</td>
<td>4</td>
</tr>
<tr>
<td>\omega_1, \omega_2, \omega_3</td>
<td>Weight of three loss terms</td>
<td>4</td>
</tr>
<tr>
<td>F</td>
<td>Feature map after inner product</td>
<td>4</td>
</tr>
<tr>
<td>F</td>
<td>Feature map after the Performer</td>
<td>4</td>
</tr>
<tr>
<td>G</td>
<td>The sliding window within D</td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>Moiré edge map (predicted)</td>
<td>4, 5</td>
</tr>
<tr>
<td>L</td>
<td>Overall loss function</td>
<td>4, 5</td>
</tr>
<tr>
<td>L_{pixel}</td>
<td>Per-pixel loss</td>
<td>4, 5</td>
</tr>
<tr>
<td>L_{dir}</td>
<td>Moiré pattern direction loss</td>
<td>4, 5</td>
</tr>
<tr>
<td>L_{dis}</td>
<td>Moiré pattern distribution loss</td>
<td>4, 5</td>
</tr>
<tr>
<td>f, \vartheta</td>
<td>MoireDet and its network weights</td>
<td>4, 5</td>
</tr>
<tr>
<td>v_1, v_2</td>
<td>Relative scaling coefficients</td>
<td>5</td>
</tr>
<tr>
<td>s_1</td>
<td>Binary class of moiré image identification</td>
<td>5</td>
</tr>
</tbody>
</table>
screen. For each camera pose, we intentionally vary the phone cameras, screen resolution and the background including the colour, transparency and scale of different natural images (see Fig. 3 (right)). Note that we only consider LCD (Liquid Crystal Display) screens in this study since CRT (Cathode Ray Tube) screens are rarely used nowadays, though some simple stripe patterns could be captured [41], [42].

We observe four factors that visibly influence moiré pattern shape, location and intensity, i.e. camera pose, camera mode, screen resolution and camera-screen distance (see Fig. 4). In terms of moiré pattern colours, even though the screen shows a white background to the human eye, the captured moiré patterns still contains colours. This is expected since each pixel on the screen is represented by a few (e.g. red, blue, green, in the case of Bayer CFA) sensors. As a result, moiré patterns and their colours are caused by the interference between the colour filter array of the camera and the sub-pixel layout of the screen. In Fig. 3 (right), we also find that the perceived moiré pattern colours (especially brightness) are slightly different on different backgrounds. In other words, moiré pattern colours are not robust to displayed background. It is worth mentioning that our observations about moiré pattern shape, intensity and colour are also in accordance to that identified in a recent work by He et al. [43].

The setup in Fig. 3 (left) can also be used to collect moiré image pairs. To ensure the consistency of moiré patterns, the selected moiré layer and moiré image pair are normally adjacent frames. We finally use the same top-left and bottom-right points on the selected pair to crop out their black borders. Though moiré patterns in the generated moiré layer are not aligned pixel-wise with the moiré image due to imperceptible vibrations from the environment, they are still visually close enough to pass as the ground truth in our experiments.

B. Collection

Motivated by the observations above, we record real-world moiré patterns of a pure white screen using a mobile phone, as shown in Fig. 5 (a). The rational behind our strategy is that: (1) This is more convenient process to mimic various camera poses and camera-screen distances in real-time; (2) A pure white screen has the least effect on distorting the moiré pattern colours [2]; (3) It is easier to generate synthetic mixtures using moiré layers on a white background. To ensure the coverage of various patterns, in addition to multiple phone and screen models, the distance (30–90 cm) and pose (yaw and pitch: [−60°, 60°]) between them are also randomly changed during acquisition. For each recorded video, frames are uniformly extracted as original moiré layers (Fig. 5(b)). In practice, even when the inner screen area is required to be always captured, it is still difficult to visually identify moiré patterns in some frames due to blurring and coinciding frequencies (frequency between camera and monitor sensors are close enough such that the moiré patterns disappear). For this, we filter these frames based on the accumulation of moiré patterns (Fig. 5(c)). Specifically, let $\mathbf{M} \in [0, 255]^{m \times n \times 3}$ denote an original moiré layer with size $[m', n']$. We first apply a convolution on $\mathbf{M}$ using a typical edge detection kernel $[−1, −1, −1; −1, 8, −1; −1, −1, −1]$. After converting and clipping (outcome shown in Fig. 6), the segmented moiré layer $\mathbf{M} \in [0, 255]^{m' \times n'}$ contains moiré-related stripes, ripples and curves of $\mathbf{M}$. Our clipping process is applied by empirically removing the textures within 10% of the area close to the moiré layer boundary. This is motivated by the fact that, (1) the screen frame could be accidentally captured due to an unstable
hand (Fig. 5(a)), and that (2) moiré patterns are relatively weaker at areas closer to the moiré layer boundary. This process can efficiently remove noises and preserve qualified moiré layers. Finally, the binary selection of an original moiré layer \( s_M \) is determined based on Eq. 1:

\[
s_M = \begin{cases} 
1 & \text{if } \sum_{i=1}^{m'} \sum_{j=1}^{n'} \tilde{M}(i, j) \geq \lambda_c \\
0 & \text{otherwise.}
\end{cases}
\]  

(1)

where \( \lambda_c \) is an empirical threshold for a fixed phone-screen combination \( c \) (different phone-screen combinations result in different range of moiré pattern densities). In Fig. 5(d), we find that moiré patterns in the selected \( M \) are visually distinct and promising.

The rationale behind \( \tilde{M} \) is that high-frequency components of \( M \) are considered in this stage. This is because in \( M \), moiré patterns are high-frequency components with fine-grained stripes, ripples, and curves that contribute to moiré effects. As such, applying the linear filter \([-1, -1, -1; -1, 8, -1; -1, -1, -1]\) on \( M \) can effectively remove the background noise and only keep the high-frequency components. Theoretically, moiré patterns are not always high-frequency distortions but also could be in (or influenced by) low-frequency, such as background colors (see Section III-A). Though low-frequency distortions contribute much lesser towards the moiré effect, they should be considered to ensure the realness of moiré images. For this, our data collection is conducted via two steps: (1) high-frequency distortion collection using a pure white screen, since it has the most negligible effect on distorting the fine-grained stripes, ripples, and curves that contribute to moiré effects. (2) Introducing low-frequency distortions via a pixel-wise multiplication of the background and the collected moiré layer (Eq. 2 in Section III-C). Thus, both high- and low-frequency distortions are properly preserved in our training set.

C. Generation

For a given candidate background \( B \in [0, 255]^{m \times n \times 3} \) and an original moiré layer \( M \) with \( s_M = 1 \), the synthetic moiré image \( I \in [0, 255]^{m \times n \times 3} \) and correlated moiré edge map \( D \in [0, 255]^{m \times n} \) should be generated to construct a training triplet \( D = \{I, B, D\} \).

1) Synthetic Moiré Image: In our framework, \( I \) is simply generated using the pixel-wise multiplication of the background and the transformed moiré layer:

\[
I = \frac{\text{Tra}(M, t)}{255} \odot B.
\]

(2)

where \( \text{Tra}(\cdot) \) is the perspective transformation operation. \( t \) denotes the perspective transformation matrix to calculate the destination mask of \( M \) on a rectangle area of size \([m, n]\). In practice, \( t \) is empirically changed to stretch, squeeze, rotate, shear and reflect \( M \), thereby imitating different camera poses and camera-display distances. Note that if the transformed \( M \) becomes smaller than \( B \), we still multiply them to generate synthetic \( I \). This is because it is normal for moiré patterns to exist regionally with clear borders, e.g. “half screen” example in Fig. 1 and “centered panel” example in Fig. 19. To improve the diversity of moiré layers, the black region (uncovered by the transformed \( M \)) in the destination mask is filled by randomly selecting from the following options: (a) 0, (b) mean moiré value, and (c) random number \( \in [0, 255] \).

Eq. 2 is motivated by the fact that multiplication can preserve both texture and colour features from \( M \). For instance, even with \( M \) collected from a pure white screen, its background could appear darker due to lighting conditions, reflections and distortions from environment, and possibly the equipment. Theoretically, if a pixel in the background \( B \) is relatively darker, the new pixel in the synthetic mixture \( I \) remains darker after multiplication with \( M \). This is because moiré patterns in \( M \) are high-frequency components with relative light colors (mostly gray). Thus, moiré patterns in close-to-white regions become more obvious (large-scale moiré noises), while in textural regions (e.g. complicated objects) these patterns are relatively weakened (small-scale moiré noises). This is exactly similar to real-world moiré effects in camera-captured screen images.

Fig. 7 presents a set of backgrounds (first row) and their phone-captured screen images (second row) with the setup provided in Fig. 3 (left). We can observe that these images become darker with clear moiré effects. We also collected moiré layers (when the background becomes pure white in the setup) to generate synthetic moiré images \( I \) based on Eq. 2. We find that the ground truth and the synthetic one (third row) are visually similar, particularly moiré pattern shape and intensity. However, we also observe that some synthetic images produced weakened pattern colours, e.g. the leftmost column in Fig. 7. This is because the moiré pattern and background image colours are slightly altered due to colour adjustment by the smartphone. Since automatic white balance is a common built-in setting, we intentionally keep it to reflect reality. To verify, observe the third column of Fig. 7: as the background image is close to white (similar to moiré layer), moiré patterns in the synthetic and ground truth images are visually the same. In Section V-B, our quantitative evaluation proves the viability of Eq. 2, as it shows that the mean L2 error between the synthetic and ground truth pairs is only 0.0996 (the best among 5 methods).

2) Moiré Edge Map: The moiré edge map \( D \) is mainly used to represent the shape and intensity of moiré patterns in
I. In our framework, \( D \) is generated by:

\[
D = \text{Tra}(\tilde{M}, t), \tag{3}
\]

where \( \tilde{M} \) is the segmented moiré layer (Fig. 6). Similar to Eq. 2, \( t \) is the perspective transformation matrix to project the segmented moiré layer \( \tilde{M} \) to a zero matrix with size \([m, n]\). Thus, \( D \) have both high-level geometrical features of moiré patterns, with fine-grained stripes, ripples and curves that are correlated to the shape, location and intensity of moiré effects.

The rationale behind Eq. 3 is that we simplify the moiré pattern detection to edge detection, making this complicated problem solvable. Moreover, we intentionally turn the coloured moiré patterns into \( D \) (grey) since detecting moiré pattern shape, location and intensity are our main objectives, which are not related to moiré pattern colours (see observations in Section III-A and verifications in [43]). Besides, as moiré pattern colours are correlated to background images, sensors and poses, involving this factor could make the detection problem unnecessarily complicated and difficult to be generalized. Finally, experiments in Section V-D and V-E show that the detected edge maps can effectively improve existing methods for moiré image identification and restoration tasks. Fig. 8 shows examples of synthetic moiré images of the same \( B \) but different \( M \). We find that the shape, location and intensity of moiré patterns are accurately represented by \( D \).

Our method has two advantages: (1) High quality in terms of pixel-level alignment and variety of moiré patterns. For instance, we can use the method in Fig. 5 to conveniently collect various real-world moiré patterns. The collected moiré patterns can be easily augmented with the function defined in Eq. 3. We can also use Eq. 2 to extract training triplets that are strictly aligned at the pixel level. These conveniences cannot be directly achieved by the aforementioned methods. (2) High efficiency and practical. After the original moiré layer collection in Fig. 5 (a), the remaining steps for training triplets generation are fully automatic. Besides, the original moiré layer collection method is straightforward and can be easily extended to different smartphones and cameras.

IV. MoireDet

Given a moiré image \( I \), our proposed MoireDet \( f \) translates its moiré edge map \( \hat{D} \) using a single network \( f(I; \theta) = \hat{D} \), where \( \theta \) is the network weights. \( f \) is trained on the dataset \( \mathcal{D} = \{(I, B, D)\} \). Note that in practice only \( \{(I, D)\} \) are employed for training \( f \). \( \{B\} \) is used for comparison experiments in Section V.

A. Network Architecture

The general idea of \( f \) is to encode both low-level texture and high-level context features of moiré patterns for the estimation of \( \hat{D} \). The rationale behind this is that moiré effects in digital photos can only be observed in the “macro-image” while caused by fine-grained textures in the “micro-image” (see Fig. 6). For this, as compactly presented in Fig. 9, \( f \) is composed of three encoders:

1) High-Level Encoder: We employ ResNet18 [46] as the backbone network, connected with two BiFPN layers [44] to repeatedly encode moiré pattern features. Since BiFPN allows top-down and bottom-up multi-scale feature fusion, it can efficiently encode high-level moiré context features (see Fig. 10). Though BiFPN outputs also contain limited low-level features, it is still too coarse for fine-grained textures [47].

2) Low-Level Encoder: We employ the earlier layers of ResNet18 (first two blocks) with attention [48] from BiFPN outputs to enhance the low-level features arising from stripes, ripples and curves that contribute to moiré effects. Since the attention from BiFPN contains rich high-level context features, the earlier layers in ResNet18 can better locate and capture these necessary structural features. We denote the combined feature map after inner product of the two as \( \hat{F} \). Intuitively, \( \hat{F} \) contains both low-level texture and high-level context features of \( I \). We then employ the Performer block [45] to
enhance the global features of $\hat{F}$ so as to reduce the influence of regional noise from non-moiré patterns. Performer is a Transformer-type architecture [49] that has proven to be robust and efficient, with only linear space and time complexity. Particularly, it uses the Fast Attention Via positive Orthogonal Random features (FAVOR+) mechanism, which leverages on new methods for approximating softmax and Gaussian kernels. As a result, it provides unbiased or quasi-unbiased estimation of the attention matrix, uniform convergence and lower variance in the approximation. We denote the feature map after the Performer as $F$ and it has the same size as $\hat{F}$. Since all pixel pairs in $\hat{F}$ are involved in the Performer, $F$ inherently contains more global features and is less sensitive to non-moiré noises such as object contours and moiré-like textures. Fig. 11 depicts moiré edge maps of a non-moiré image with and without the Performer block.

Fig. 11. Comparison of estimated moiré edge map $\hat{D}$ of a non-moiré image with and without the Performer block.

3) Spatial Encoder: We develop a convolutional encoder with adaptive kernels $(5 \times 5$ size) from $F$ to encode low-level moiré patterns in $I$ with respect to different spatial regions. In other words, our specific adaptive kernel only calculates the activation for a particular region rather than a sliding window. This is motivated by the fact that both the shape and intensity of fine-grained moiré patterns are regionally different (see Fig. 8). Weights in adaptive kernels are calculated with two $1 \times 1$ convolutions on $F$. Thus, the kernels in our method are not only inherently sample-specific, but also location-specified for each sample. Noted that our method is different from the dynamic kernels in [50] and [51]. Particularly, in [51] each sample is encoded into a one-dimensional feature vector and then convoluted with a sample-specific kernel. In [50] a dynamic kernel is also only sample-specific since it is calculated based on simple convolutions with hyperbolic tangents from the original image. On the other hand, our method is more adaptive (sample-specific) and can better encode low-level moiré patterns in $I$ (location-specific). Feature maps from the three encoders are finally concatenated to estimate the moiré edge map $\hat{D}$ after undergoing a further two $1 \times 1$ convolutions.

B. Loss Functions

Our loss function $L$ is composed by three terms to encode differences between $\hat{D}$ and its ground truth $D$: a per-pixel loss $L_{\text{pixel}}$, a moiré pattern direction loss $L_{\text{dir}}$ and a moiré pattern distribution loss $L_{\text{dis}}$. The overall loss is:

$$L(\theta) = \omega_1 L_{\text{pixel}}(\theta) + \omega_2 L_{\text{dir}}(\theta) + \omega_3 L_{\text{dis}}(\theta).$$

where we empirically set $\omega_1 = 1$, $\omega_2 = 0.8$ and $\omega_3 = 0.8$ to balance the weight of each term. Specifically, the initial weights are set to 1.0, then empirically updated based on the observed converging speed during the training process. For instance, we observed the per-pixel loss has the slowest converging than the other two. To balance the contributions of three losses, we intentionally tone down the weights of the direction and distribution losses. In the future, we will try some automatic methods [52], [53], [54] to further optimize their weights. Fig. 12 presents the general idea of these three terms:

1) Per-Pixel Loss: Per-pixel loss $L_{\text{pixel}}$ measures the difference of $\hat{D}$ and $D$ in pixel intensities. We employ Smooth L1 function to calculate errors between each pixel pair. Loss $L_{\text{pixel}}$ is the average value of all errors $d(\hat{D}, D)$:

$$L_{\text{pixel}}(\theta)=d(\hat{D}, D) = \frac{1}{m \times n} \sum_{i=1}^{m} \sum_{j=1}^{n} s(\hat{D}(i, j), D(i, j)).$$

where $s(\cdot)$ is the Smooth L1 error between the pixel pair $(\hat{D}(i, j), D(i, j))$. However, $L_{\text{pixel}}$ does not accurately address the quality of moiré patterns in terms of direction and distribution. Hence, two additional terms are introduced.

2) Direction Loss: Direction loss aims to evaluate the quality of $\hat{D}$ by comparing the directions of fine-grained stripes, ripples and curves with $D$. This is understandable since high-level moiré effects in a camera-captured screen image highly depend on the direction and entanglement of some small segments (see Fig. 6). As shown in Fig. 12 (right), we filter both $\hat{D}$ and $D$ with different binary kernels and calculate the difference of each filtered pair. Fig. 13 illustrates these 14 binary kernels that are developed based on statistics obtained from segmented moiré layers $M$ regarding the shape and direction of small segments [55]. We can see that each

Fig. 12. Our loss function $L$ is composed by three terms: $L_{\text{pixel}}, L_{\text{dir}}$ and $L_{\text{dis}}$. The overall loss is presented in Eq. 4.

Fig. 13. 14 binary kernels that are used to filter moiré pattern directions. Each kernel is in $7 \times 7$ size.
kernel represents a unique component of the curve and its direction. Loss $L_{dir}$ is the average value of all pairs:

$$L_{dir}(\theta) = \frac{1}{N_K} \sum_{i=1}^{N_K} d(Conv(\tilde{D}, K_i), Conv(D, K_i)). \quad (6)$$

where $Conv(\cdot)$ denotes the convolution operation, $d(\cdot)$ is similar to that in Eq. 5, $K$ is a binary kernel of $N_K$ number of kernels. In our case, $N_K = 14$.

3) Distribution Loss: As presented in Fig. 8, the shape and intensity of moiré patterns are distributed quite differently in each moiré image. Distribution loss $L_{dis}$ aims to compare the difference of moiré pattern distributions in each region. We use a $7 \times 7$ sliding window to collect and compare corresponded regions in $\tilde{D}$ and $D$ (Fig. 12 (left)). The Smooth L1 error is calculated based on the square deviation values in each pair. Loss $L_{dis}$ is the average value of all pairs:

$$L_{dis}(\theta) = \frac{1}{N_G} \sum_{i=1}^{N_G} s(\text{Var}(\tilde{G}_i), \text{Var}(G_i)). \quad (7)$$

where $\tilde{G}$ and $G$ are the matrices within the sliding window of $\tilde{D}$ and $D$, respectively. $N_G$ is the number of sliding windows in $D$ (as with $\tilde{D}$ also). $\text{Var}(\cdot)$ denotes variance of the distributions and $s(\cdot)$ is the Smooth L1 function.

C. Implementation and Discussion

We use input image sizes of $320 \times 320 \times 3$ and output $\tilde{D}$ is also of size $320 \times 320$. $\theta$ is initialized by Xavier [56]. Model $f$ is implemented using PyTorch (convolution with adaptive kernels can be implemented with the help of unfold function). We use absolute position embedding [57] in the Performer block after normalizing to within $[0, 1]$. We optimized the loss functions (in Section IV-B) using Adam [58] with a mini-batch size of 8, an initial learning rate of 3e-4, weight decay of 1e-5. We ran for a total of 100 epochs reducing the learning rate by a factor of 10 in every 30 epochs. The entire training process took around 28.6 hours on average on a NVIDIA TITAN Xp GPU. Our proposed MoireDet was trained from scratch.

<table>
<thead>
<tr>
<th>Phone</th>
<th>Camera(MP)</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xiaomi 10</td>
<td>48</td>
<td>AOC</td>
</tr>
<tr>
<td>Smartisan U1</td>
<td>13</td>
<td>AOC</td>
</tr>
<tr>
<td>Xiaomi Note3</td>
<td>12</td>
<td>TOSHIBA</td>
</tr>
</tbody>
</table>

To the best of our knowledge, there is no publicly available dataset for evaluating the moiré pattern detection task. Therefore, we create a benchmark dataset, MoireScape, consisting of synthetic image triplets $D$ and real image pairs. Specifically, the synthetic image triplet is generated by the proposed strategy in Section III. As presented in Table II, we use three different phone and display models (in total $3 \times 3 = 9$ combinations) to collect various moiré layers $M$. We also randomly selected 1,000 natural images $B$ from each of these existing datasets: COCO [59], ImageNet [60], PASCAL VOC [61] and Retail50K [62]. We note that images in recent Retail50K have the most complex background among these datasets due to its densely packed scene. In total, 18,147 different moiré layers and 4,000 natural images were collected.

A. Datasets

1) Detection: To the best of our knowledge, there is no publicly available dataset for evaluating the moiré pattern detection task. Therefore, we create a benchmark dataset, MoireScape, consisting of synthetic image triplets $D$ and real image pairs. Specifically, the synthetic image triplet is generated by the proposed strategy in Section III. As presented in Table II, we use three different phone and display models (in total $3 \times 3 = 9$ combinations) to collect various moiré layers $M$. We also randomly selected 1,000 natural images $B$ from each of these existing datasets: COCO [59], ImageNet [60], PASCAL VOC [61] and Retail50K [62]. We note that images in recent Retail50K have the most complex background among these datasets due to its densely packed scene. In total, 18,147 different moiré layers and 4,000 natural images were collected.

By varying $\theta$ in Eq. 2, we finally generate 50,000 synthetic triplets for the purpose of training (90%) and testing (10%). We also collected 500 real image pairs for evaluating moiré edge map estimation (see Fig. 14). Each pair includes a...
camera-captured screen image and its moiré edge map with the setup in Fig. 3 (left). For fair comparison, phone and display models in evaluation set are different from that of the training set, as also with the background images. In short, MoireScape has two subsets for testing (Synthetic and Real), both are challenging in three aspects: (1) regional moiré patterns in some images; (2) various real-world moiré patterns from subtle to obvious; (3) varying complexity of backgrounds.

2) Identification: The identification task aims to classify whether an image contains moiré patterns. Though Abraham et al. [40] introduced a benchmark of 1,633 images, the full dataset is still not publicly available at the time of developing this work. Earlier, Garcia et al. [10] collected a MoireFace dataset for the purpose of face-spoofing detection. Particularly, they conditionally selected 50 original face images from three actively used databases [66], [67] and then displayed them in Macbook, iPad and iPhone for camera capturing. With 12 different camera, display and distance combinations, they collected $12 \times 50 = 600$ face images contaminated by moiré patterns (positives). Together with the 50 original images (negatives), there are 650 face images in this dataset.

In practice, typical face spoofing attacks include impersonation (2D paper print, 2D/3D screen replay, 3D paper mask, 3D mannequin) and obfuscation (glasses, makeup, and tattoo). Among them, 2D/3D replay attacks using tablets and smartphones are getting popular and feasible. Though no APPs utilize moiré detection for face-spoofing yet, several research works have been published. For instance, Garcia et al. presented a moiré image identification method for face-spoofing detection [9], [10]. Furthermore, Liu et al. introduced a spoof-trace generator that disentangles several spoof-trace elements, including moiré patterns [7]. Finally, Bian et al. assessed several face anti-spoofing methods and positively verified moiré patterns’ usability [8]. The rationale behind these methods is that moiré patterns are inherently additive components of 2D/3D replay attacks on digital devices. While our proposed approach is more efficient and practical, we hope that the public release of this method will inspire and encourage the industrial and AI communities to develop more robust algorithms. For instance, one possible idea is to use moiré patterns as additional information for higher reliability of face spoofing detection.

We also employ the demoiréing datasets FHDMi [43] and MRBI [68] in our experiments. The FHDMi dataset contains 2019 image pairs for testing. Each pair contains a moiré image for demoiréing, and a moiré-free image as the ground truth. Similarly, the MRBI dataset contains 340 pairs for testing. The moiré and ground truth images are labeled as positive and negative for moiré image identification. To encourage the community to adapt moiré image identification methods for more complex environments, we organize another new dataset, MoiréIDT, of 4,000 images from different scenarios, including 2000 real moiré images (positive) and 2000 moiré-free images (negative) of different complexity of backgrounds. To evaluate the generalization of methods, the positive set not only contain camera-captured screen images, but also some natural images with moiré effects (see Fig. 15).

3) Restoration: In addition to FHDMi [43] and MRBI [68], the actively used dataset DMCNN [2] is also employed for the comparison of moiré image restoration before and after the employment of our framework. It consists of 135,000 screen-shot image pairs. Each pair contains an image contaminated with a moiré pattern and its corresponding uncontaminated reference.

B. Synthetic Image Generation

A naive and widely used approach is to simply mix $M$ and $B$ via $I = v_1M + v_2B$, where $v_1$ and $v_2$ are relative scaling coefficients ($v_1 + v_2 = 1$) to avoid overflow or image clipping [65], [69], [70], [71], [72]. However, scaling the images not only constrains $M$ and $B$ within a relatively smaller color range, but also suppresses abrupt color transitions, especially for $M$. Recently, Fan et al. [64] proposed to generate synthetic reflection image based on a heuristic and simple subtraction operation. Zhang et al. [11] enhanced this method in [64] to better approximate the physical formation of images and reflection from oblique angles. Though both methods generate various reflection effects by varying the standard deviation $\sigma$ of the Gaussian filter, they are still sub-optimal for moiré image generation. This is because $M$ is blurred by the Gaussian filter thus moiré patterns with small gradients are not well maintained in $I$. Moreover, it normally require manual tuning of $\sigma$ and $v_1$ to reduce hazing and color shifting in the background. To avoid parameters, the classic Poisson blending with mixed seamless cloning [63] could be employed since it is applied with gradient fields of $(M, B)$ to facilitate partially transparent objects like moiré patterns. However, the dark regions in $B$ become brighter in $I$ since its non-linear mixing of gradient files only picks up more salient structures.

For validation, we randomly download 20 moiré-free images from internet and collect their moiré layers and ground truth pairs. Each pair contains an image contaminated with a moiré pattern and its corresponding uncontaminated reference.

Fig. 15. Positive samples in MoiréIDT. Top: Camera-captured screen images. Bottom: Natural images with moiré effects.
most similar to camera-captured screen images. Thus, training triplets generated by our strategy are more realistic and this can benefit future work in moiré detection, reduction and removal.

C. Moiré Pattern Detection

We first train and test the proposed MoireDet using the MoireScape dataset to build baselines. Specifically, MoireDet was trained by 50,000 × 90% = 45,000 triplets and tested by two sets: 50,000 × 10% = 5,000 synthetic triplets and 500 real image pairs. Similar to Section V-B, we use the mean distance to evaluate moiré edge map D. Here D is evaluated since it not only presents the location and density of moiré patterns in I for image quality evaluation, but also provides a prior towards improving the efficiency of restoration tasks in Section V-E. As presented in Table III, we conducted a set of ablation studies to investigate the effectiveness of encoders and loss functions. We find that using all three encoders and losses (last column) achieve the best performance on both Synthetic and Real testing sets. We believe that the proposed encoders and loss functions work interdependently to capture both low-level and high-level features of moiré patterns. A more intuitive impression can be visually observed in Fig. 17 where some low-level features were missed by HLE and HLE+SE while fine-grained moiré patterns also disappeared in Pix and Pix+Dis. Regarding input size, we find that 320 × 320 has the best performance on Synthetic set and the close-to-best (only 0.0005 lower than 280 × 280) on Real set. Thus, we decided on using it for the remaining experiments considering both accuracy and training speed.

For comparisons, we adapted four networks that were designed for reflection separation (HFNet [11]), shadow detection (DSCNet [73]), blur detection (DCNN [19]) and image matting (DED [74]) to the moiré pattern detection scenario. In Table IV, HFNet and DCNN perform poorly while DSCNet and DED are better in performance but not as good as ours. We compare this visually in Fig. 18, where we observe that both low- and high-level features are not preserved in HFNet and DCNN. Though DSCNet and DED can capture high-level features, some details in the left and bottom regions are missing. Overall, MoireDet has shown to be effective at detecting moiré patterns at both levels.

D. Moiré Image Identification

We extend MoireDet for moiré image identification by simply applying Eq. 1 on D to output the binary class of I. σ = 1 means the accumulated moiré edge in D is above the threshold λc and image I is identified to be contaminated by
TABLE V
MOIRÉ IMAGE IDENTIFICATION BASELINES ON FOUR DATASETS. P AND R ARE THE PRECISION (%) AND RECALL (%) OF MOIRÉ IMAGES. ACC DENOTES ACCURACY (%).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Wavelet [40]</td>
<td>38.54</td>
<td>40.20</td>
<td>38.05</td>
<td>76.24</td>
</tr>
<tr>
<td>Peak [9]</td>
<td>74.71</td>
<td>79.90</td>
<td>76.43</td>
<td>95.27</td>
</tr>
<tr>
<td>MDCNN [40]</td>
<td>50.32</td>
<td>97.75</td>
<td>50.63</td>
<td>92.96</td>
</tr>
<tr>
<td>MoireDet</td>
<td><strong>84.13</strong></td>
<td><strong>77.90</strong></td>
<td><strong>81.60</strong></td>
<td><strong>95.88</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Wavelet [40]</td>
<td>38.54</td>
<td>76.24</td>
<td>23.00</td>
<td>42.73</td>
<td></td>
</tr>
<tr>
<td>Peak [9]</td>
<td>74.71</td>
<td>95.27</td>
<td>97.33</td>
<td>93.08</td>
<td></td>
</tr>
<tr>
<td>MDCNN [40]</td>
<td>50.32</td>
<td>92.96</td>
<td>88.00</td>
<td>82.77</td>
<td></td>
</tr>
<tr>
<td>MoireDet</td>
<td><strong>84.13</strong></td>
<td><strong>90.57</strong></td>
<td><strong>81.60</strong></td>
<td><strong>95.88</strong></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 19. Natural images with moiré effects in MoireIDT dataset and their estimated \( D \) using the proposed method.

Fig. 20. Detected moiré patterns on the MoireFace dataset [10]. The first column shows the original images, the remaining columns are captured using different phones and screen setups, as mentioned in [10].

Fig. 21. Detected moiré patterns on the FHDMI [43] (top) and MRBI (bottom) [68] testing sets captured from different screens.

moiré patterns, and vice versa. \( \lambda_c \) is only correlated with input image size and we here set it to 0.01. Results for MoireDet against three other methods are detailed in Table V.

Briefly, the method of Garcia et al. [9] identifies moiré patterns based on peak detection in the frequency domain of bandpass-filtered moiré images, Abraham et al. [40] developed a wavelet frequency thresholding approach and also a network MDCNN. As expected, the Peak method achieves the highest recall and accuracy in MoireFace since its parameters were specially tuned for such scenarios. The MDCNN method is easily over-fitted to moiré images, as it achieves the highest recall in most datasets. Moreover, it has low precision (around 50%), as it only focus on the coarse-grained features of moiré patterns, and ignores the fine-grained stripes, ripples, and curves.

Differently, MoireDet is more generalized and achieves the highest precision on most datasets, even though it was trained without moiré patterns from their domains (e.g., small displays in [9]). In the context of anti-spoofing applications, both recall and precision are important. Notably, recall is vital to ensure system security, while precision is essential to ensure system usability and user experience. A little recall failure for anti-spoofing is still acceptable. For instance, users can lock out a device or account for a specific period of time, so they do not have infinite retries on the system. Besides, users may not mind trying to unlock multiple times to circumvent the low recall problem (in most cases, it is due to poor lighting conditions and/or unqualified face poses), while having a high precision is crucial to prevent theft and unauthorized access.

While our method achieves higher precision on most datasets and for this context, it is closer to the requirements of real-world applications. In practice, we can also improve the recall by enriching the original moiré layers with the method in Fig. 5. Our preliminary experiments on this idea showed that the recall of MoireFace can be improved from 97.33 to 98.57 by simply collecting more moiré layers from phone screens. Overall, our proposed MoireDet is with the most stable performance on the four datasets, has a better balance of precision and recall, and yields the best performance on the F1 score.

In summary, our proposed MoireDet is well generalized and can detect moiré patterns in both natural images (see Fig. 19) and captured images from different screens (see Fig. 20 and 21). Thus, MoireDet is more balanced across all metrics compared to MDCNN [40]. Generally, our detection results could be particularly meaningful for some spoofing detection applications in smart-retail and smart-home
scenarios – we find these directions are viable avenues for further exploration.

### E. Moiré Image Restoration

Theoretically, moiré pattern detection results can also be employed to improve the efficiency of other tasks such as moiré image restoration. This is because detecting these patterns can provide priors to better target moiré regions. To demonstrate this, three recent moiré removal methods are employed on DMCNN [2] and MoireScape datasets – 1) He et al. [5]: MopNet; 2) Yang et al. [24]: HRDN (High-Resolution Demoire Network); 3) Xu et al. [25]: a fractally stacked network called AFN (Attentive Fractal Network). Here the Synthetic subset in MoireScape is used since the original and moiré images are aligned pixel-wise. To fully use the shape, region and intensity information, we adapted \( \tilde{I} \) as the fourth channel alongside \( I \) as input to the networks. The widely-used metrics for image quality evaluation, PSNR (Peak Signal-to-Noise Ratio) and SSIM (Structural SIMilarity) are adopted, and we report the comparisons in Table VI. On MoireScape, all three networks achieved around 1.43 (PSNR) and 0.018 (SSIM) improvements with the addition of \( \tilde{I} \). However, the improvement on DMCNN is less obvious than in MoireScape. The main reason is that some moiré patterns in DMCNN are accurately detected. For instance, the leftmost image is distorted by scanned line noise and MoireDet still detects it robustly. We also find that detected moiré patterns in the second image (swimwear, marked in red) contain some edges from the background (on the right side). This is caused by both fine-grained moiré patterns and accidental light beams that are moiré-like. To verify this, observe carefully the top-left part of the swimwear image: the fine-grained moiré patterns are very faint in this region, hence the corresponding region in the moiré edge map is nearly empty even though some light beams (which is part of the original image) are visible. The fine-grained moiré patterns on the top-right are more intense, and thus visible moiré edges are detected. However, the model is misled by light beams in this region since they are similar to moiré patterns at the coarse-grained level. To solve this problem, negative samples that are intuitively similar to moiré patterns should be added to the training set.

To further evaluate the usability of MoireDet, we demonstrate the demoiréing performance using different training and testing sets. For instance, the AFN [25] and our proposed MoireDet are trained with the DMCNN and MoireScape datasets, respectively. The best results in each dataset are marked in bold.

To further explore the usefulness of MoireDet, we demonstrate the demoiréing performance using different training and testing sets. For instance, the AFN [25] and our proposed MoireDet are trained with the DMCNN [2] and the synthesized MoireScape datasets, then their corresponding trained models are applied to the FHDMi [43] and MRBI [68] testing sets. While the models were trained and testing with different datasets, the comparisons are more fair. In Table VII, even with different training sets, the demoiréing performance still improves with \( I+\tilde{D} \).

Even so, as presented in Fig. 22, we can clearly see that MoireDet generalizes quite well as most visually distinct moiré patterns in DMCNN are accurately detected. For instance, the leftmost image is distorted by scanned line noise and MoireDet still detects it robustly. We also find that detected moiré patterns in the second image (swimwear, marked in red) contain some edges from the background (on the right side). This is caused by both fine-grained moiré patterns and accidental light beams that are moiré-like. To verify this, observe carefully the top-left part of the swimwear image: the fine-grained moiré patterns are very faint in this region, hence the corresponding region in the moiré edge map is nearly empty even though some light beams (which is part of the original image) are visible. The fine-grained moiré patterns on the top-right are more intense, and thus visible moiré edges are detected. However, the model is misled by light beams in this region since they are similar to moiré patterns at the coarse-grained level. To solve this problem, negative samples that are intuitively similar to moiré patterns should be added to the training set.

To further evaluate the usability of MoireDet, we demonstrate the demoiréing performance using different training and testing sets. For instance, the AFN [25] and our proposed MoireDet are trained with the DMCNN [2] and the synthesized MoireScape datasets, then their corresponding trained models are applied to the FHDMi [43] and MRBI [68] testing sets. While the models were trained and testing with different datasets, the comparisons are more fair. In Table VII, even with different training sets, the demoiréing performance still improves with \( I+\tilde{D} \).

### Table VI

<table>
<thead>
<tr>
<th></th>
<th>DMCNN</th>
<th>MoireScape</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td>20.3</td>
<td>0.738</td>
</tr>
<tr>
<td>MopNet [5]</td>
<td>27.44</td>
<td>0.890</td>
</tr>
<tr>
<td>HRDN [24]</td>
<td>28.47</td>
<td>0.860</td>
</tr>
<tr>
<td>AFN [25]</td>
<td>27.00</td>
<td>0.830</td>
</tr>
</tbody>
</table>

### Table VII

<table>
<thead>
<tr>
<th></th>
<th>FHDMi</th>
<th>MRBI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td>17.97</td>
<td>0.703</td>
</tr>
<tr>
<td>MopNet [5]</td>
<td>19.19</td>
<td>0.72</td>
</tr>
<tr>
<td>HRDN [24]</td>
<td>19.29</td>
<td>0.72</td>
</tr>
<tr>
<td>AFN [25]</td>
<td>19.84</td>
<td>0.72</td>
</tr>
</tbody>
</table>

Fig. 22. Detected moiré patterns using our proposed MoireDet on the DMCNN dataset.
edges are not detected for the first image in Fig. 22. The main reason is that our training set does not contain such moiré patterns (similar to the chessboard). In fact, most moiré images in the DMCNN dataset were collected via relatively old smartphones and screen models (submission: November 2017, publication: May 2018). Particularly for smartphones, the automatic white balance and image beautification algorithms could vary in different models, influencing the moiré pattern features. As a result, some rare moiré patterns that only occur in low-resolution and old models are not covered by our proposed MoireScape dataset. Even so, the trained MoireDet (using MoireScape training set) is generalized well and can detect most moiré patterns in the DMCNN testing set.

F. Limitations

Fig. 23 presents some failure cases in MoireFace, MoireScape and MoireIDT datasets. We find that MoireDet is limited in two aspects: (1) too fine or too coarse patterns, and (2) too dark background. Particularly, MoireDet sometimes struggle to detect moiré patterns in camera-captured phone images (MoireFace) in Fig. 23 (a) due to the fact that certain moiré patterns (and their high-level features) are not distinctive on phone and tablet screens. In Fig. 23 (b),(c) and (d), we find a different scenario where MoireDet struggles with to handle moiré patterns with single high-level features. Since (d) was intentionally captured with big camera-display distance while fine-grained features are barely covered, D is nearly empty. In a similar case of Fig. 23 (e), an image with rather dark background clouds the moiré patterns resulting in nearly empty D values as well.

VI. CONCLUSION AND FUTURE WORKS

Moiré pattern detection is a relatively less-studied problem that deserves more attention due to its importance to computational photography and image restoration. It is also particularly beneficial as a prior task to a wide range of related applications. In this article, we introduce a framework for moiré pattern detection which includes a practical strategy for preparing high quality training triplets and a novel neural network method consisting of three key encoders for moiré edge map estimation. We demonstrated that our framework can be further adopted for two other tasks, moiré image identification and restoration. Along with that, two new datasets were proposed, to encourage further work in the community. In addition to the said practical scenarios (spoofing detection, recapture detection, guideboard analysis, fashion design, and media production), we also believe that this work can be extended to detect other image patterns that are intuitively similar to moiré pattern features: (1) Integration of both high- and low-frequency noises, (2) more obvious globally, and (3) dynamic effects while rescaling the image. Thus, we will extend MoireDet to more practical scenarios. We will also propose two stage methods (detection and restoration) to improve moiré image restoration performance. The data and code associated with this paper can be found at https://github.com/cong-yang/MoireDet.

ACKNOWLEDGMENT

The authors would like to thank Liang Sang and Yulu Zhang from Clobotics for their great help on data collection and discussion during the experiments.

REFERENCES


Cong Yang received the Ph.D. degree in computer vision and pattern recognition from the University of Siegen, Germany, in 2016. He was a Postdoctoral Researcher at the MAGRIT Team, INRIA, France. He worked scientifically and led the Computer Vision and Machine Learning Teams at Clobotics and Nanjing Horizon Robotics Technology Company Ltd. He has been an Associate Professor with Soochow University, since 2022. His main research interests include computer vision, pattern recognition, and their interdisciplinary applications.

Zhenyu Yang received the bachelor’s and master’s degrees in civil engineering from Southeast University, Nanjing, China, in 2016 and 2019, respectively. He was a Computer Vision Engineer at Nanjing Horizon Robotics Technology Company Ltd. He has been a Computer Vision Engineer with Pony.ai, since 2021. His research interests include computer vision, machine learning, and their applications.

Yan Ke received the bachelor’s degree in computer science, the master’s degree in electrical and computer engineering, and the Ph.D. degree in computer science in 2008, all from Carnegie Mellon University. He spent eight years at Microsoft leading the Bing Entity Understanding Group until 2016. He is the Co-Founder and the CTO of Clobotics—a computer vision company serving the retail and wind industries. He is an expert in computer vision, data mining, machine learning, and distributed systems. He has published in top international computer vision conferences, such as IEEE CVPR and ICCV, and holds eight U.S. patents.

Tao Chen received the B.Sc. degree in mechanical design, manufacturing and automation, the M.Sc. degree in mechatronic engineering, and the Ph.D. degree in mechatronic engineering from the Harbin Institute of Technology, Harbin, China, in 2004, 2006, and 2010, respectively. He is a Visiting Scholar with the National University of Singapore in 2018. He is currently a Professor with the School of Future Science and Engineering, Soochow University, Suzhou, China. His main research interests include MEMS, sensors, and actuators.

Marcin Grzegorzek received the D.Eng. degree in pattern recognition from the University of Erlangen–Nuremberg in 2007. He worked scientifically at the Queen Mary University of London and the University of Koblenz and Landau. From 2010 to 2018, he was an Assistant Professor of pattern recognition at the University of Siegen. He is a Professor of medical informatics with the University of Luebeck. His research interests include medical data science.

John See received the B.Eng., M.Eng.Sc., and Ph.D. degrees from Multimedia University, Malaysia. He was a Visiting Research Fellow at Shanghai Jiao Tong University, China. He is an Associate Professor with the School of Mathematical and Computer Sciences, Heriot-Watt University (Malaysia). He regularly publishes in reputable journals and major computer vision conferences. His research interests include computer vision and image processing, particularly in the emerging fields of facial micro-expressions, affective computing, computational aesthetics, and deep learning. He was a recipient of the Belt and Road Initiative Young Scientist Exchange Fellowship. He is an Associate Editor of IEEE TRANSACTIONS ON MULTIMEDIA (T-MM), EURASIP JIVP, and Frontiers in Signal Processing.