A convolutional neural network-based joint detection and localization spatiotemporal scheme for process control through speckle pattern imaging

Sabahno, Hamed; Khodadad, Davood

Published in:
Computers & Industrial Engineering

DOI:
10.1016/j.cie.2025.111538

2025

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

Computers & Industrial Engineering 210 (2025) 111538

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

A convolutional neural network-based joint detection and localization spatiotemporal scheme for process control through speckle pattern imaging

Hamed Sabahno *, Davood Khodadad

Department of Applied Physics and Electronics, Umeå University, 90187 Umeå, Sweden

ARTICLE INFO

Keywords:
Convolutional neural network
Process control
Control charts
Image data
Speckle pattern analysis
Monte Carlo simulation

ABSTRACT

This paper presents a novel convolutional neural network (CNN)-based control chart for detecting localized and subtle speckle shifts (displacements) in the images. To enhance detection accuracy, especially in detecting localized and simultaneous shifts, we design a custom CNN architecture and utilize a training methodology. Moreover, instead of inputting the entire images, we divide them into equal-sized grids and process them as two-channel inputs: one containing the shifted grids and the other capturing the difference between shifted and reference grids. A maximum probability-based control scheme is developed to concurrently detect shifted images and localize shifted regions within an image. To the best of our knowledge, this is also the first spatiotemporal scheme that simultaneously performs detection and localization, while conducting image-based process control. We perform simulation studies under two main scenarios: (i) spatial domain analysis and (ii) frequency domain analysis. The control chart's performance is evaluated across various grid sizes, shift magnitudes, and shifted areas. Additionally, we demonstrate our scheme's capability through several real-time monitoring examples. Comparative analysis with the existing methods showed that our CNN-based control chart significantly outperformed the existing approach, particularly in detecting small global shifts, localized shifts, and simultaneous shifts.

1. Introduction

Control charts serve as a fundamental tool in statistical process monitoring (SPM), designed to identify assignable causes of variation within a process and differentiate them from natural fluctuations. Traditionally, control charts have been constructed using statistical methods. However, in recent years, there has been a growing shift towards machine learning (ML) models due to their enhanced capabilities. While nowadays ML models are being extensively used in many industrial and manufacturing processes (e.g., Nguyen-Ngoc et al., 2024; Luong et al., 2025; Guo et al., 2025; Nguyen et al., 2025; Kocabıçak et al., 2025), several studies including those by Niaki and Abbasi (2008), Hosseinifard et al. (2011), Mohammadzadeh et al. (2023), Sabahno and Amiri (2023), Sabahno and Niaki (2023), and Yeganeh et al. (2024) have demonstrated that ML-based control charts outperform traditional statistical control charts in certain scenarios. Machine learning-based control charts provide a flexible alternative to traditional statistical methods when working with complex, nonlinear, or high-dimensional data, such as images. While they forgo some of the statistical rigor of classical approaches, they enable powerful modeling of subtle and spatially distributed changes that are difficult to capture analytically.

In addition, there has been increasing interest in adapting control chart methodologies to analyze image data and spatiotemporal variations. Megahed et al. (2011) conducted a comprehensive review of control charting developments using image data until 2011. Later, Megahed et al. (2012) introduced a control chart based on the generalized likelihood ratio (GLR) to monitor image data, assessing its performance using the dice similarity coefficient test. Koosha et al. (2017) proposed a nonparametric profile monitoring approach, utilizing wavelet transformation for feature extraction and monitoring approximation coefficients with a GLR-based control chart. Their method was found to surpass that of Megahed et al. (2012) in specific cases. Koosha et al. (2022) further extended this work by introducing a two-dimensional wavelet-based framework. Similarly, Khodadadi et al. (2024) enhanced prior studies by applying contourlet transformation to image data in conjunction with a GLR control chart. Other notable contributions include Amirkhani and Amiri (2020), who developed a p-value-based control chart for image data monitoring, incorporating

* Corresponding author at: Department of Applied Physics and Electronics, Umeå University, 90187, Umeå, Sweden.
E-mail addresses: hamed.sabahno@umu.se (H. Sabahno), davood.khodadad@umu.se (D. Khodadad).

https://doi.org/10.1016/j.cie.2025.111538
Received 14 May 2025; Received in revised form 12 September 2025; Accepted 16 September 2025
Available online 20 September 2025

Dunnett's test for fault localization and a maximum likelihood estimator for change point detection. Building on this, Yeganeh et al. (2024) improved the p-value-based chart by introducing a partitioning ensemble control chart that integrates machine learning, ensemble learning, and image partitioning techniques. Additionally, Sabahno and Khodadad (2025) explored and introduced control charting through speckle pattern analysis, which is proven to be very effective on featureless surfaces as well as detecting subtle shifts.

As industrial systems grow increasingly complex, conventional image processing techniques face limitations, particularly in handling high-frequency patterns, noise, and subtle variations. One area where these traditional methods fall short is speckle pattern analysis. Speckles, random granular light patterns caused by the interference of coherent light scattered by rough or irregular surfaces, are crucial in non-destructive testing (NDT) and optical metrology (Hecht and Zajac, 1974; Khodadad, 2016; Sirohi, 2020). Speckle analysis is widely used for measuring displacements and detecting physical changes such as deformation, strain, and flow velocity. Its applications span various industries, including NDT (Pang et al., 2020; Khodadad et al., 2023; Zhang et al., 2024), optics and metrology (Torre et al., 2016; Qureshi et al., 2023; Chao et al., 2024; Smolovich et al., 2024; Zhou et al., 2024), medical imaging and diagnostics (Iwase et al., 2015; Abdel-Nasser et al., 2024), material and surface science (Jamali et al., 2023), fluid mechanics (Raffel, 2007), and agriculture (Zdunek, 2014; Khodadad and Sabahno, 2025). The ability to detect localized and subtle changes within an image is critical for high-precision applications such as material testing, semiconductor manufacturing, and surface inspection. However, conventional image processing methods struggle to identify micro-defects, high-frequency variations, and subtle texture shifts essential for process control and defect detection. Speckle pattern analysis provides a more effective alternative by leveraging high-frequency components inherent in speckle patterns, enabling the detection of even the smallest shifts and imperfections. Detecting subtle shifts and micro-defects is essential in many high-precision industrial contexts. For example, in semiconductor manufacturing, nanometer-scale surface misalignments can lead to circuit failure and yield loss. In aerospace, small deformations or strains on fuselage or wing structures can signal early fatigue that, if left undetected, may lead to catastrophic failure. In mechanical systems, such as gearboxes or rotating machinery, local surface wear often begins as minute, localized texture changes are undetectable by traditional vision systems but diagnosable through speckle pattern variations. These examples highlight the practical necessity of detecting small, distributed shifts in production and inspection pipelines.

Furthermore, Convolutional Neural Networks (CNNs) have significantly advanced image processing by improving feature extraction and pattern recognition. Their hierarchical architecture, which employs convolutional layers to learn spatial hierarchies from raw image data, has led to remarkable improvements in tasks such as object detection, image segmentation, and enhancement (Simonyan and Zisserman, 2015; He et al., 2016; Krizhevsky et al., 2017; Xu et al., 2022; Shen et al., 2023; Choi et al., 2024). Within speckle metrology, CNNs have proven effective in enhancing speckle pattern analysis, automating detection and quantification, and improving displacement and deformation measurements. Recent studies have demonstrated how CNNs refine speckle pattern reconstruction and extract meaningful features from complex interference patterns, thereby enhancing non-destructive testing and metrological applications. Deep learning-based image segmentation and feature extraction techniques have been particularly valuable in analyzing speckle images, facilitating more accurate and reliable measurements in speckle metrology (Nguyen et al., 2021; Montresor et al., 2022; Kwon et al., 2023). Although CNNs have been previously used for speckle pattern analysis, they have not been used for speckle image-based control charting and process monitoring.

As previously mentioned, Sabahno and Khodadad (2025) introduced control charting based on speckle images. They used Fourier magnitude spectrum differences to construct their control chart. They also use a memory-type control chart to speed up small shifts detection. Moreover, they used an additional p-value analysis for localization.

In this paper, we will significantly improve the work of Sabahno and Khodadad (2025). Among existing works, only Sabahno and Khodadad (2025) propose a control chart specifically tailored to speckle pattern images. As such, we compare our proposed method against this method as the most relevant and structurally similar benchmark. We use a custom omnibus CNN model with two-channel inputs to construct the control charts for detection, and at the same time, localize the shifted areas. Our scheme is based on dividing the image into equal-sized grids. We construct the control chart based on the maximum out-of-control probabilities among the grids of the speckle image. Although image gridding itself is common, the novelty here is inputting individual grids into the CNN instead of using the whole image, and the integration of gridwise CNN outputs into a single MPr control statistic that simultaneously signals and localizes faults within the same sampling. We tailore the CNN architecture, and also its training methodology, so we can quickly detect both small and large speckle shifts (displacements). Unlike their work which was only performed in the frequency domain, we will explore both spatial and frequency domains. Also, to better show our scheme's superiority, we do not use a memory-type control chart, as they did. In addition, we use an industrial product example (a gear in this study) for our numerical analysis and evaluated the charts' performance under different individual or simultaneous shifts, grid sizes, and affected areas. This study is designed as a proof-of-concept to evaluate the feasibility of using CNN-based control charts for detecting and localizing subtle shifts in speckle patterns under controlled conditions. Our approach complements traditional SPC by addressing use cases (such as image-based surface inspection, where statistical assumptions are violated and subtle texture shifts must be detected and localized in real-time).

Moreover, traditional control charts, such as Shewhart and EWMA, offer strong statistical foundations but rely on assumptions like normality, independence, and linearity that often do not hold for image-based or high-dimensional process data. In contrast, machine learning (ML)-based methods — while lacking theoretical guarantees — can model complex input structures and learn subtle spatiotemporal patterns, including localized shifts. Our CNN-based framework embraces this tradeoff: it uses empirical calibration to define control thresholds and enables localized detection in cases where classical SPC tools underperform. The approach provides a data-driven alternative particularly suited to modern industrial applications such as surface defect detection, tool wear monitoring, and structural health inspection — all of which demand smart, image-based quality control strategies.

In summary, the main contributions of this study are:

- A novel grid-wise CNN-based control scheme that jointly performs detection and spatial localization of subtle structural shifts in speckle textures.
- Evaluation of the framework in both spatial and frequency domains, demonstrating flexibility across signal representations.
- A data-efficient input construction using only two channels per grid: the shifted grid and a local feature map (e.g., spatial difference or frequency residual).
- A full pipeline for process-level and grid-level decision-making, enabling interpretable shift localization without requiring synthetic labels at test time.

This paper is structured as follows: In Section 2, we describe the speckle pattern analysis method. Section 3 contains the proposed CNN-based process control scheme which includes designing the CNN architecture and its training methodology, accompanied by detection and localization methods. Section 4, contains extensive simulation studies accompanied by real-time process monitoring and localization examples. In Section 5, we present our concluding remarks and suggestions

for future works.

2. Speckle pattern analysis

Speckles are random granular interference patterns that occur when coherent light (e.g., laser light) is scattered by a rough or irregular surface. The scattered waves interfere with one another, producing a seemingly random distribution of bright and dark spots. This phenomenon, known as "speckle," is caused by the following process: (i) a coherent light source such as a laser illuminates the surface; (ii) the surface scatters the light in various directions; (iii) the scattered waves interfere constructively (bright spots) and destructively (dark spots); (iv) the resulting interference pattern appears as a dense, granular texture. These speckle patterns are highly sensitive to minute surface displacements, making them suitable for precision sensing and quality control. Fig. 1 depicts the speckle pattern creartion process.

In this paper, we analyze the speckle patterns in both spatial and frequency domains. In the absence of a real speckle pattern, we can simulate one. We begin by generating a reference speckle pattern from a grayscale image, which is converted to the frequency domain using a random phase mask and the Fourier transform. For how a speckle pattern can be generated we refer interested readers to Sabahno and Khodadad (2025). For transforming the speckle pattern into the frequency domain, we use the Fourier transform, as it is commonly used in speckle pattern analysis. In addition, instead of performing these transformations on the whole speckle pattern image, in this paper we first divide the image into equal-sized grids and then perform relevant operations on each grid separately.

Before the Fourier transform, we use windowing with a Blackman window to enhance the stability of the results and to avoid artifacts caused by abrupt transitions at the grid's edges to reduce spectral leakage due to edge discontinuities between grid boundaries (Harris, 1978). Although CNNs offer partial translation invariance, this windowing operation is essential for improving frequency-domain feature quality rather than for training stability or shift generalization.

$$w(z) = a_0 - a_1\cos\left(\frac{2\pi z}{T-1}\right) + a_2\cos\left(\frac{4\pi z}{T-1}\right), \tag{1}$$

where $w(z)$ is the value of the window function at index z, that is the index along one of the rows or columns of the section, T is the total number of samples in the window, $a_0 = 0.42$, $a_1 = 0.5$, and $a_2 = 0.08$.

This formula generates the coefficients of the Blackman window, which are then multiplied element-wise with $I(x,y)$, the intensity of the grids at spatial coordinates of (x,y), to apply the windowing effect. Therefore, we have:

$$f_1(x,y) = I_1(x,y).w(x).w(y), \tag{2}$$

$$f_2(x,y) = I_2(x,y).w(x).w(y), \tag{3}$$

where $f_1(x,y)$ and $f_2(x,y)$ are the intensity functions of the reference and deformed (shifted) grids affected by the Blackman window, respectively.

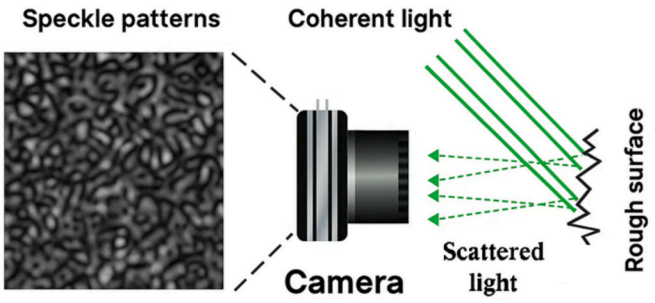

Fig. 1. Creartion of speckle patterns.

When we divide an image into equal-sized grids and want to analyze each grid without losing information, applying a Blackman window to each grid can help in the following ways, even if we want to perform the image analysis in the spatial domain (referring to the first scenario in Section 4.1.1):

1. Reducing edge effects

Normally, when an image is divided into grids, information at the boundaries can be lost or discontinuities can appear between adjacent grids. A Blackman window smoothly tapers the pixel values toward the edges of the grid, reducing sudden transitions that could lead to artifacts.

2. Maintaining spatial Continuity

If a shift occurs in the image, a harsh grid-based segmentation without windowing might miss subtle displacements. The Blackman window ensures that neighboring grids still retain some overlap in terms of spectral content, which can help in detecting shifts.

The Fourier transform of the spatial domain grid $f(x,y)$ is:

$$F(u,v) = \iint_{-\infty}^{\infty} f(x,y).e^{-2\pi i(ux+vy)}dxdy, \tag{4}$$

where $F(u,v)$ is the Fourier transform of the grid, and u and v are the spatial frequency coordinates.

After Fourier transformation, we apply a high-pass filter to each grid to suppress low-frequency components that mainly reflect global illumination and background structure. Subtle shifts and micro-textural changes primarily affect the high-frequency spectrum of speckle patterns, as shown in speckle metrology studies. By emphasizing high-frequency content, we enhance the model's sensitivity to discriminative local shifts while suppressing irrelevant background variations.

To do so, we first obtain the Fourier magnitude spectra F_1 and F_2 of the reference and deformed grids, respectively, and apply a high-pass filter $H\ (u,v)$. The enhanced Fourier magnitude spectra of the reference and deformed grid become:

$$F'_1(u,v) = |F_1(u,v)|.H(u,v), \tag{5}$$

$$F'_2(u,v) = |F_2(u,v)|.H(u,v), \tag{6}$$

where $H\ (u,v)$=$(1 - X\ (u,v))(2 - X\ (u,v))$ and $X\ (u,v) = \cos\ (\pi u).cos(\pi v)$.

3. Proposed CNN-based process control scheme

In this section, we propose a CNN-based method for shift detection and the localization of the shifted areas in a speckle pattern. Before presenting our proposed CNN architecture and its training methodology, which are carefully tailored and its hyperparameters carefully selected and tuned to achieve our goal, we encourage readers to read the Appendix: Fundamentals of CNNs.

3.1. CNN architecture and training methodology

For the goal of detecting small to large localized speckle shifts, we design the following CNN architecture, as described in Table 1. The architecture and hyperparameter values were determined through extensive experimentation and iterative refinement, involving evaluation under diverse conditions to ensure that the CNN-based approach reliably performs both detection and localization.

The input layer in our study is four-dimensional and includes: [grid height, grid width, channels, grids]. We generate equal numbers of in- and out-of-control images as the CNN input. As in Sabahno and Khodadad (2025), we only consider translational displacements and generate the in-control (IC) images using displacements generated by a bivariate normal distribution $N(\boldsymbol{\mu} = \mathbf{0}, \boldsymbol{\Sigma} = \mathbf{I})$. For the out-of-control (OC) images, we generate the shifts using $N(\boldsymbol{\mu} = \mathbf{2}, \boldsymbol{\Sigma} = \mathbf{I})$. To train the CNN, shifts are synthetically applied to localized regions in simulated speckle images. This provides full control over ground truth, enabling systematic evaluation of the model's shift sensitivity and localization

Table 1
The proposed CNN architecture.

Layer/Block Type	Description
Input Layer	*−Two-channel inputs (shifted + difference grids)*
Convolutional Block 1	*− Convolution layer (8 filters, 3 × 3, dilation = 1, padding = same)*
	− Batch Normalization layer
	− LeakyReLU layer (α = 0.1)
Convolutional Block 2	*− Convolution layer (16 filters, 3 × 3, dilation = 1, padding = same)*
	− Batch Normalization layer
	− LeakyReLU layer (α = 0.1)
Convolutional Block 3	*− Convolution layer (32 filters, 3 × 3, dilation = 2, padding = same)*
	− Batch Normalization layer
	− LeakyReLU layer (α = 0.1)
Fully Connected layers	*− Dense layer with 32 neurons*
	− LeakyReLU layer (α = 0.1)
	− Dropout layer (0.5)
	− Dense layer with 2 neurons
Output Layers	*− Softmax layer*
	− Classification Layer (IC / OC)

capability. As mentioned before, we divide the images into equal-sized grids in this research. To better train the CNN about the IC and OC images, and also to be able to not only use the CNN model for detecting, but for localization of the shifted areas of the speckle pattern, instead of the images, we enter each grid separately into the model. In addition, the grids are entered in two channels. In the first channel, the shifted grids (shifted IC or OC grids) are entered. In the second channel, the difference between the shifted grids and the reference grids is also entered, enabling the CNN to learn about the reference image as well. Note that, as it will be further discussed in Section 4, the spatial and frequency domain data are handled as separate input sources in two independent experimental scenarios in this study. The CNN is trained either on raw spatial grids or on log-magnitude frequency-domain grids obtained after windowing, FFT and high-pass filtering. The two input types are not fused or used as multi-channel inputs within the same network. This design enables isolated evaluation of detection and localization performance within each domain.

The reference image is the original speckle image before applying any IC or OC shifts on it. Note that, we could have made the input layer with three layers, or two layers including the reference and shifted grids instead, but after extensive ablation studies during our development phase we discovered the current combination to be the most effective one, similar to inputting the grids instead of the images which we previously discussed. We have two classes (labels) in this study: IC (labeled 0) and OC (labeled 1) and we dedicate them to each grid depending on whether they belong to an IC or OC image. Moreover, in IC images, all the grids are IC, and in OC images, all the grids are OC. However, we found it to be most effective to apply the same IC shift (generated by $\mathrm{N}(\boldsymbol{\mu} = \mathbf{0}, \boldsymbol{\Sigma} = \mathbf{I})$) to all the grids of an IC image, but apply different shifts (each generated by $\mathrm{N}(\boldsymbol{\mu} = \mathbf{2}, \boldsymbol{\Sigma} = \mathbf{I})$) to each grid of an OC image. In other words, all the grids of an IC image shift uniformly (we first generate and apply the IC shift to the whole image and then divide it into grids, in this case), but all the grids of an OC image shift differently (we first divide the image into grids, and then generate and apply the OC shift to each grid separately, in this case). This is based on the logic and assumption that an IC shift affects the whole image uniformly and an OC shift can occur in any area of the image. This is also very helpful in detecting simultaneous shifts that occur in different areas of the image.

In all the convolutional layers, we have chosen small filter sizes of 3 × 3 to capture fine-grained details (better for small shifts), preserve local displacements, and maintain more spatial resolution. We also use the same padding in all the layers to keep all the input features, even small displacements, in the output. In the two first convolution layers, the dilation factor of one is used, which is actually the default value. However, for the third convolution layer, the dilation factor of two is used which means the kernel is spread out by skipping 1 pixel between sampled points. We designed the convolution layers as such so the first two layers focus on fine, local texture variations, making sure small speckle shifts are detected and the third layer (with dilation = 2) enhances spatial awareness, allowing the model to pick up slightly larger contextual changes while retaining fine details. Table 2 outlines the summary of the CNN architecture mostly focusing on the convolution layers and their purpose.

In addition, to avoid neurons becoming inactive and ensure gradient flow for negative inputs, we use another function for activation (instead of the popular method mentioned in the appendix), and it is called leaky ReLU:

$$Leaky\,ReLU(x) = \begin{cases} x & if\ x > 0 \\ \beta x & if\ x \leq 0 \end{cases}, \tag{6}$$

where β is the leak factor, and we set its value to 0.1. This helps prevent dead neurons in neural networks by maintaining a small gradient even for negative values.

Note that, since we deal with highly detailed speckles in this paper, we do not use a pooling layer. Pooling layers were omitted to maintain spatial resolution across all layers. Since the objective is to detect subtle, localized variations in texture (e.g., small shifts in speckle structure), pooling could dilute or eliminate essential local features. To expand the receptive field without losing spatial precision, we instead apply dilation in the final convolutional layer.

We specifically design the training methodology, by considering faster computation, overfitting avoidance, and small shifts detection.

For the optimizer, we use Adam because it is an ideal choice for small gradient updates. Speckle displacement detection requires precision, and Adam ensures stable updates without large fluctuations.

For training the CNN, we set the maximum number of epochs to 20 and use a piecewise learning rate scheduling as:

$$\eta_t = \eta_0 \cdot \gamma^{\left\lfloor \frac{t}{T} \right\rfloor}, \tag{7}$$

where η_t is learning rate at step t, η_0 (set to 0.0005) is the initial learning rate, γ (set to 0.5) is the drop factor, and T (set to 10) is the drop period. The learning rate is halved every 10 epochs, allowing coarse learning at the start and finer adjustments later, making it suitable for precise displacement detection. In addition, we choose a small initial learning rate (0.0005) to avoid overshooting fine shifts.

For training stability and regularization, we use L2 regularization and gradient clipping. L2 regularization (weight decay) adds a penalty term as described in Equation 15. We set $\lambda = 0.005$ in this research to preserve small displacements while preventing overfitting.

Also, to prevent gradient explosion, we clip gradients as:

Table 2
Convolution layers in the proposed CNN architecture.

Layer Type	Key Parameters	Purpose
Input	grid size × grid size × 2	Takes in a 2-channel grayscale image (grid)
Convolution 1	3 × 3, 8 filters	Detects edges, small displacements
Convolution 2	3 × 3, 16 filters	Detects complex small shifts
Convolution 3	3 × 3, 32 filters, dilation = 2	Captures larger displacement patterns
Fully Connected 1	32 neurons, dropout 0.5	Converts extracted features into a compact representation
Output	2 neurons, softmax	Predict the probability of the two classes (IC and OC)

$$g_t = \frac{g_t}{\max\left(1, \frac{\|g_t\|}{c}\right)}, \tag{8}$$

where g_t is the raw gradient, and c is the threshold (set to 1). Gradient clipping avoids exploding gradients, which can happen if small shifts cause unstable updates. It ensures stable updates by preventing excessively large gradients that could disrupt small displacement learning.

The mini-batch size is set to 32 and at the beginning of each epoch, we shuffle the dataset to avoid pattern memorization. Shuffling ensures that mini-batches contain different combinations of samples in each epoch, improving generalization. We chose a small mini-batch size for small displacements, but not too small to avoid noise.

Furthermore, validation is used to assess the model's performance on unseen data during training, helping to prevent overfitting. We keep 10 % of the dataset for validation, and 5 % for testing. We run validation every 50 iterations (mini-batches). This means the model trains for 50 mini-batches and then evaluates the validation set without updating weights. This helps catch overfitting early, prevent data leakage, provide a true performance measure, optimize the learning rate dynamically, ensure that regularization is effective, and detect instabilities in training.

In summary, our architecture uses three convolutional layers with 3 × 3 filters, which are optimal for preserving local shift sensitivity. Smaller kernels allow the network to learn fine-grained texture distortions that characterize subtle speckle shifts. The number of filters (8, 16, 32) increases with depth to capture progressively complex spatial features. A dilation of 2 is used in the third convolutional layer to expand the receptive field, allowing the model to encode larger-scale correlations. Dropout (0.5) is applied after the fully connected layer to prevent overfitting, tuned via validation accuracy.

3.2. Detection and localization

As mentioned in the previous section, the proposed CNN architecture outputs two probabilities (IC and OC probabilities). To design the control chart for process monitoring, and thereafter localization of the shifted areas of the image, we use the OC probability of each grid. We define:

$$MPr = \max_j Pr_j, \tag{9}$$

where MPr is the maximum of OC probabilities among all the image grids, and Pr_j is the OC probability of grid j. Then, we use MPr for constructing the control chart. It means that at each sampling, the MPr value (computed using Equation 9) will be compared against the control limits for signal detection. The next step in designing a control chart is determining the probability of false alarm (false alarm rate), α. We follow Sabahno and Khodadad (2025) and set its theoretical value to 0.05, in this research. The last step in designing a control chart is determining the control limits values. Due to the nature of our problem, in which only the high OC probabilities are crucial, we only use the UCL (upper control limit) in this research. In a Shewhart-type control chart, we have $UCL = \frac{1}{\alpha} = 20$. However, in other control charts (like ours) in which the normality and independence assumptions cannot be confirmed, we can adjust the UCL value, so we get a fixed predetermined ARL value (average run length) which is the main performance measure of the control charts (Sabahno and Eriksson, 2024). ARL is the average number of samples (images) taken before an OC signal is detected by the control chart.

Fig. 2 shows how the proposed monitoring scheme operates at each sampling. With the control chart set for process monitoring, one image is taken at each smapling. Then, the image is divided into equal-sized grids and each grid is fed into the CNN in two channels (its current form and the difference between its current form and the reference from). Then, the CNN computes each grid's OC probability (Pr), and their maximum is also computed (MPr). If $MPr > UCL$, the process is declared as out-of-control (detection), and the grids with their $Pr > UCL$ are flagged as significant (localization). Note that the same UCL will be used for both detection and localization purposes, even though its value is adjusted using MPr to achieve a pre-determined ARL, prior to online process monitoring.

4. Numerical analyses

To perform our numerical analyses, we use the same gear's image (Fig. 3) as used in Sabahno and Khodadad (2025). The main goal is to evaluate the proposed process control scheme under different scenarios and cases. In addition, to be able to compare our proposed AI-based scheme with the magnitude spectrum difference-dependent one proposed in Sabahno and Khodadad (2025), we keep the simulation environment in both papers the same. Although their scheme was only applied in the frequency domain, we apply our CNN-based scheme in both spatial and frequency domains. Moreover, while they used a memory-type scheme, which inherently increases the chart's performance especially in detecting small to moderate shift sizes, to better demonstrate the power of our AI-based scheme, we use a memory-less scheme.

In addition, the same speckle pattern of 600 × 600 pixels is also used, as shown in Fig. 4.

Note that, while the experiments in this section are based on simulated patterns, they provide a structured and replicable platform for validating detection/localization accuracy under known shift conditions.

As mentioned in Section 2, we use several preprocessing techniques in this study, applied individually to each grid or patch of the image. To illustrate the visual effect of each step, we extract a 100 × 100 patch from the top-left corner of the image in Fig. 4 and apply the preprocessing stages in sequence.

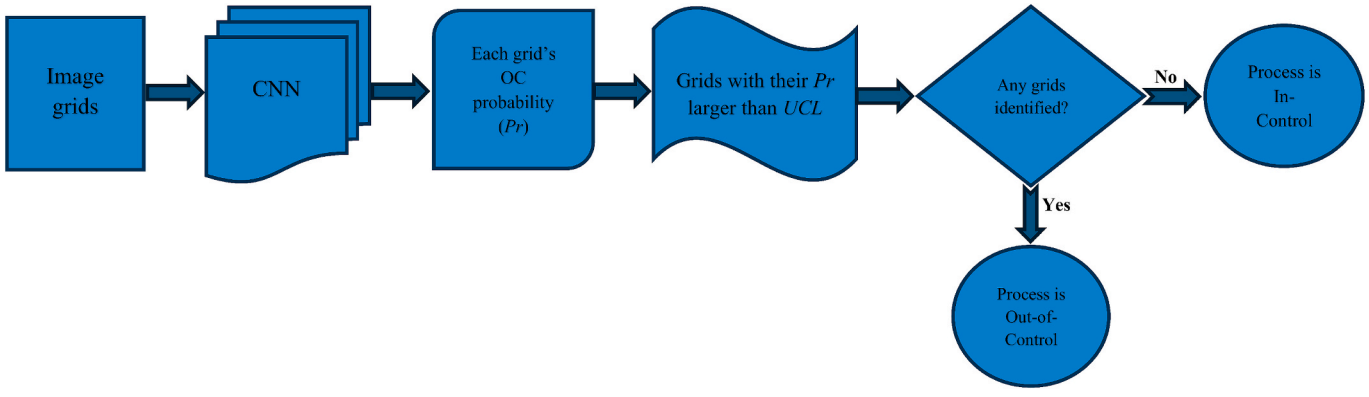

Fig. 2. Process monitoring scheme at each sampling.

Fig. 3. A gear's image.

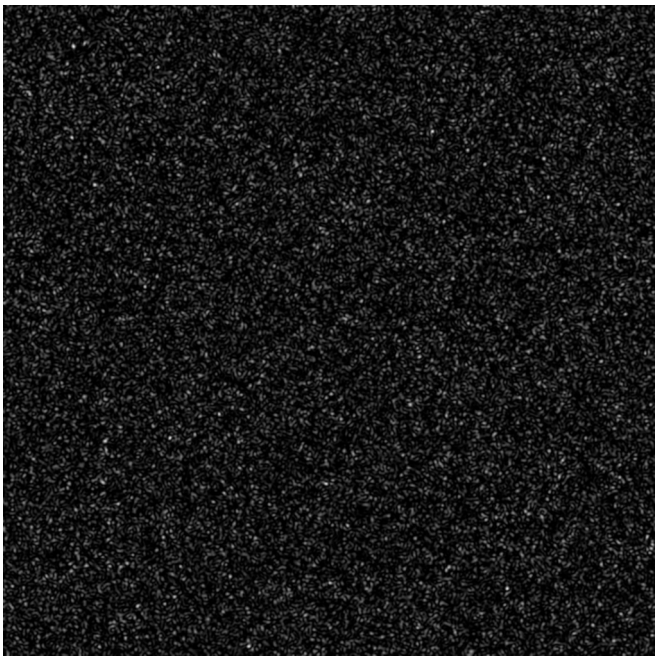

Fig. 4. The gear's speckle pattern.

Fig. 5 shows the results of this process using two parallel paths—with and without Blackman windowing. The top row presents the unprocessed path: the original raw intensity patch, its FFT magnitude spectrum (log-scaled for visualization), and the high-pass filtered spectrum obtained by applying a frequency-domain filter directly to the FFT of the raw patch. The bottom row shows the windowed path: the Blackman-windowed patch (which is created by applying the windowing fucntion on the original patch), its FFT magnitude spectrum, and the spectrum after applying the same high-pass filter.

Other than showing the effect of each preprocessing operation, this comparison highlights the role of windowing: the unwindowed FFT spectrum exhibits stronger edge effects and potential spectral leakage, while the windowed version offers a smoother, cleaner frequency response. The high-pass filter then enhances higher frequency components, which correspond to subtle texture variations critical for shift detection and defect identification.

4.1. Control chart and signal detection

In this section, we evaluate the charts' performance in two main scenarios of spatial and frequency domains, and in three gridding cases of 100×100, 50×50, and 30×30 for each scenario. For our analyses in this section, we use a powerful laptop equipped with an NVIDIA GeForce RTX 4090 GPU (16 GB VRAM), an Intel i9 14900HX processor, and a 32 GB system RAM. Smaller grid sizes produce more grids and therefore increase per-sample compute and memory roughly in proportion to the grid count. For example, for a fixed 600×600 image, 30×30-pixel grids yield $20 \times 20 = 400$ grids in the image, while 100×100-pixel grids yield $6 \times 6 = 36$ grids (more than $11 \times$ difference). Consequently, grid size should be chosen to trade off localization granularity versus available compute/latency; if finer localization is required but compute is constrained, multi-scale (coarse-to-fine) refinement, selective re-evaluation of candidate regions, or model compression/quantization should be considered. To adapt to new speckle statistics we recommend transfer learning using a small calibration set (freeze early layers and fine-tune later layers or the final head).

To conduct our simulation studies, we adjust the *UCL* value for each chart (case-scenario) to obtain the ARL value of 20, under 5000 IC runs. This will make it easier for us to perform simulation studies, due to the extremely heavy and computationally demanding of the image/grids-based scheme. After estimating the *UCL* in each case by targeting an ARL of 20, we also computed the empirical false alarm rate (α) to independently evaluate in-control performance. Note that, as mentioned before, since the normality and independence assumptions cannot be confirmed in our scheme, we expect the empirical false alarm rate not to be equal to its theoretical value 0.05.

It is clear that because in each scenario and also each gridding case the inputs are different, different CNN models should be trained for each as well. In both scenarios, we generate 400 images for each gridding case. In addition, half the images in each case are generated in-control (shifts are generated by $N(\mathbf{0},\mathbf{I})$) and the other half are generated out-of-control (shifts are generated by $N(\mathbf{2},\mathbf{I})$), and the IC or OC shifts are applied as was detailed in Section 3.1. The shift magnitude (e.g., $N(\mathbf{2},\mathbf{I})$) is applied in the spatial domain and represents a translational displacement in pixels. This is a standardized, unitless measure within the image frame. The conversion to physical units (e.g., microns) can be determined by the calibration of the specific imaging system (e.g., microns/pixel), which is an application-dependent parameter.

To evaluate the charts' performance, artificial shifts are applied to five different random areas of the 600×600 speckle pattern as shown in Fig. 6.

The shifts (the columns of the following tables) are only applied to the mean of the normal distribution, and it is assumed that the variances (as well as covariances) remain unchanged throughout the monitoring procedure. Therefore, the shift cases in the columns are respectively created using: $N(\mathbf{0.2},\mathbf{I})$, $N(\mathbf{0.5},\mathbf{I})$, $N(\mathbf{0.9},\mathbf{I})$, $N(\mathbf{2},\mathbf{I})$, $N(\mathbf{3},\mathbf{I})$, $N(\mathbf{5},\mathbf{I})$, and $N(\mathbf{10},\mathbf{I})$.

Note that, regardless of whether the analysis is performed in the spatial domain (first scenario) or the frequency domain (second scenario), all shifts are applied to the intensities in the spatial domain (Fig. 4), with the necessary transformations carried out afterward.

Furthermore, the ARL (average run length) and SDRL (standard deviation of run length) are used to measure the charts' performance. However, comparisons are only performed based on the ARL values and the SDRL values are only reported for additional details. We also report ARL/20 values in the tables, so it would be easier to compare to the chart of Sabahno and Khodadad (2025).

As shown in all the tables in this section (Tables 3-11), the shifts are

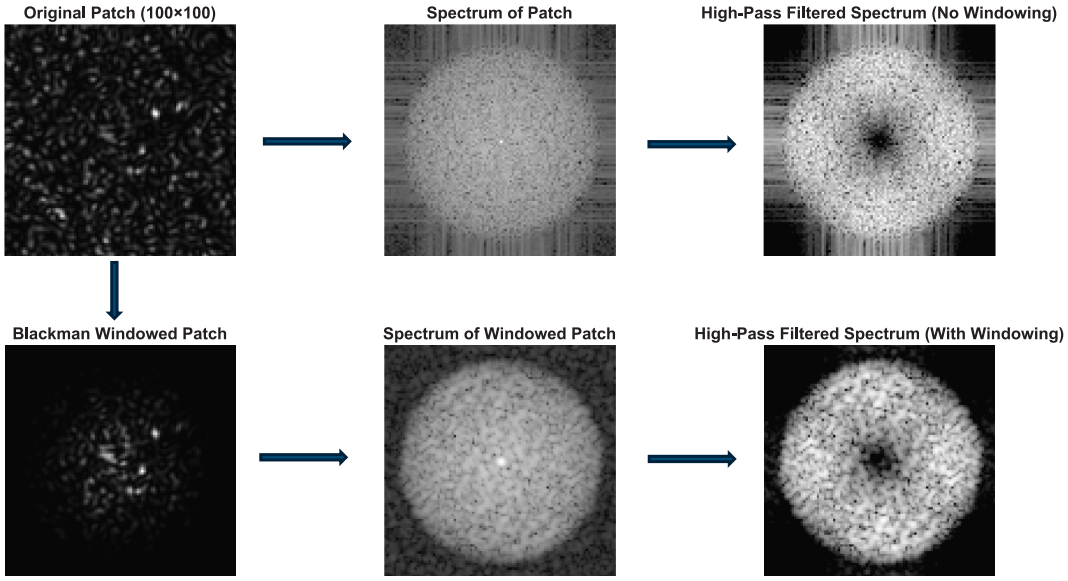

Fig. 5. Illustration of the role of windowing in the preprocessing pipeline, using a 100 × 100 image patch.

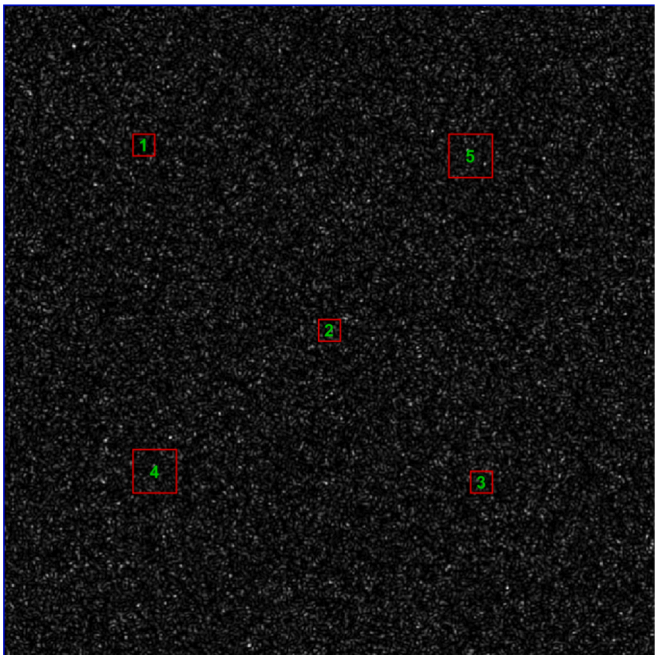

Fig. 6. The shifted areas in the speckle pattern.

applied in three stages: first, each affected area is shifted solely; next, all five areas are shifted simultaneously; and finally, the entire monitored pattern is uniformly shifted. In addition, the simulation is conducted as follows: when an area is shifted by a value drawn from N(***x***,**I**), the rest of the pattern is shifted equally by a value generated from N(**0**,**I**) to simulate a natural movement during the image capturing. Additionally, when all five areas are shifted, each area's shift size is independently drawn from N(***x***,**I**), resulting in different shifts across areas, while the rest of the pattern is again shifted equally by a value from N(**0**,**I**).

4.1.1. First Scenario: Speckle image in the spatial domain

In the first scenario, the image in Fig. 4 is divided into equal-sized grids and will be analyzed in the spatial domain. In addition, each gridding case is analyzed in two sub-scenarios. In sub-scenario 1, the grids are directly fed into the CNN, while in sub-scenario 2, they are fed into the CNN after the windowing is applied to them.

Table 3
ARL, SDRL, ARL/20 values of the control chart under different mean shift sizes and in different areas of the image, with the 100 × 100 grids size and in the spatial domain without windowing.

Shifted area	Shift size 0.2	0.5	0.9	2	3	5	10
First area	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC
Second area	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC
Third area	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC
Fourth area	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC
Fifth area	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC
All five areas	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC
Whole image	13.87, 13.17, 0.69	7.97, 7.45, 0.4	4.08, 3.52, 0.2	1.28, 0.61, 0.06	1.01, 0.12, 0.05	1.00, 0.00, 0.05	1.00, 0.00, 0.05

In the first scenario, we apply our CNN-based scheme in the spatial domain. We also consider two sub-scenarios for the first scenario. In sub-scenario 1, the speckle patterns ($I_1(x,y)$ for the reference image and $I_2(x,y)$ for the IC and OC images) are directly fed into the CNN (as inputs) after gridding and without performing any of the operations outlined in Section 2. In sub-scenario 2, however, we apply the windowing function (Equation 1) to each grid before inputting them (we input $f_1(x,y)$, Equation 2, and $f_2(x,y)$, Equation 3, in this sub-scenario instead of $I_1(x,y)$ and $I_1(x,y)$ used in the previous sub-scenario).

The objective in this scenario is evaluating the model's ability to detect and localize subtle shifts in speckle images using spatial (raw intensity) features alone. This simulates real-time process monitoring directly from pixel-level variations.

As mentioned before, we use simulated speckle patterns generated from grayscale base images. The images are partitioned into grid sizes of 100 × 100, 50 × 50, and 30 × 30. Controlled shifts are introduced either globally or locally. Each scenario is tested over 5000 Monte Carlo repetitions to estimate *UCL*s and ARLs. Tables 3-8 show the ARL performance for various grid sizes and shift magnitudes.

Table 4
ARL, SDRL, ARL/20 values of the control chart under different mean shift sizes and in different areas of the image, with the 100 × 100 grids size and in the spatial domain with windowing.

Shifted area	Shift size 0.2	0.5	0.9	2	3	5	10
First area	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC
Second area	Same as IC	Same as IC	Same as IC	15.96, 15.50, 0.8	11.30, 10.32, 0.57	5.64, 5.17, 0.28	4.86, 4.40, 0.24
Third area	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC
Fourth area	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC
Fifth area	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC
All five areas	Same as IC	Same as IC	Same as IC	15.02, 13.99, 0.75	12.07, 11.80, 0.6	5.67, 5.17, 0.3	5.03, 4.71, 0.25
Whole image	13.25, 12.74, 0.66	6.35, 5.92, 0.32	3.09, 2.58, 0.15	1.17, 0.44, 0.06	1.01, 0.08, 0.05	1.00, 0.00, 0.05	1.00, 0.00, 0.05

Table 5
ARL, SDRL, ARL/20 values of the control chart under different mean shift sizes and in different areas of the image, with the 50 × 50 grids size and in the spatial domain without windowing.

Shifted area	Shift size 0.2	0.5	0.9	2	3	5	10
First area	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	11.23, 10.31, 0.56
Second area	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	4.92, 4.63, 0.25
Third area	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC
Fourth area	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	17.23, 16.29, 0.86
Fifth area	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC
All five areas	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	4.06, 3.52, 0.2
Whole image	12.77, 12.41, 0.64	7.21, 6.47, 0.36	3.80, 3.34, 0.19	1.30, 0.63, 0.06	1.02, 0.14, 0.05	1.00, 0.00, 0.05	1.00, 0.00, 0.05

Case 1: 100 × 100 grids.

In this case, we divide the image into equal-sized grids of size 100 × 100 (which makes 36 grids in the image), as shown in Fig. 7.

In the first sub-scenario, the grids are directly inputted into the CNN without the windowing function applied to them. For this sub-scenario, after training the CNN, the validation accuracy is computed as 99.24 % and the test accuracy is computed as 99.58 %. Then, as previously mentioned, using the developed CNN model, for this case and using 5000 simulation runs, we estimate the *UCL* to achieve the desired ARL of 20, as 0.922. We have also computed the empirical false alarm rate (under again 5000 simulation runs) as 0.0499, which is very close to its theoretical value of 0.05.

Table 3 shows the control chart performance under this gridding case, without windowing. As it can be seen in this table, the control chart under this gridding case and this sub-scenario is unable to detect localized shifts and the chart's performance in this case-scenario is similar to the IC performance, under all shift sizes and in all shifted areas. However, when the whole image shifts uniformly, the chart performs very well in detecting shifts and its performance improves as the shift size increases.

Moreover, when we compare this chart to the chart designed by Sabahno and Khodadad (2025) under the same gridding (its Table 3), both charts perform badly in detecting localized shifts, but our chart performs faster in detecting small shifts when the whole image is shifted

Table 6
ARL, SDRL, ARL/20 values of the control chart under different mean shift sizes and in different areas of the image, with the 50 × 50 grids size and in the spatial domain with windowing.

Shifted area	Shift size 0.2	0.5	0.9	2	3	5	10
First area	18.71, 18.30, 0.94	16.74, 15.98, 0.84	13.45, 12.87, 0.67	6.53, 5.85, 0.33	5.09, 4.38, 0.25	2.26, 1.70, 0.11	1.01, 0.11, 0.05
Second area	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC
Third area	15.09, 14.41, 0.75	14.48, 13.85, 0.72	12.25, 11.44, 0.61	6.33, 5.83, 0.32	8.40, 7.72, 0.42	3.42, 2.87, 0.17	1.00, 0.00, 0.05
Fourth area	9.24, 8.63, 0.46	6.19, 5.38, 0.31	3.29, 2.74, 0.16	1.19, 0.47, 0.06	1.01, 0.12, 0.05	1.01, 0.09, 0.05	1.00, 0.00, 0.05
Fifth area	8.65, 8.15, 0.43	5.63, 5.15, 0.28	2.94, 2.38, 0.15	1.19, 0.48, 0.06	1.01, 0.09, 0.05	1.00, 0.02, 0.05	1.00, 0.00, 0.05
All five areas	5.31, 4.85, 0.27	3.33, 2.78, 0.17	1.83, 1.22, 0.09	1.02, 0.14, 0.05	1.00, 0.00, 0.05	1.00, 0.00, 0.05	1.00, 0.00, 0.05
Whole image	11.77, 10.95, 0.59	5.80, 5.34, 0.29	2.83, 2.18, 0.14	1.13, 0.39, 0.06	1.00, 0.07, 0.05	1.00, 0.00, 0.05	1.00, 0.00, 0.05

Table 7
ARL, SDRL, ARL/20 values of the control chart under different mean shift sizes and in different areas of the image, with the 30 × 30 grids size and in the spatial domain without windowing.

Shifted area	Shift size 0.2	0.5	0.9	2	3	5	10
First area	11.15, 10.59, 0.56	7.84, 7.42, 0.39	4.73, 4.25, 0.24	1.75, 1.14, 0.06	1.30, 0.64, 0.05	1.10, 0.34, 0.05	1.05, 0.24, 0.05
Second area	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	2.91, 2.31, 0.14
Third area	15.09, 14.09, 0.75	15.77, 15.13, 0.79	17.14, 16.35, 0.86	19.31, 19.01, 0.96	19.26, 18.51, 0.96	18.93, 18.00, 0.95	18.86, 18.41, 0.94
Fourth area	8.86, 8.39, 0.44	8.15, 7.85, 0.4	7.22, 6.82, 0.36	5.51, 4.86, 0.27	2.95, 2.36, 0.15	1.23, 0.52, 0.06	1.07, 0.27, 0.05
Fifth area	8.44, 7.61, 0.42	8.28, 7.69, 0.41	8.20, 7.78, 0.41	10.40, 9.80, 0.52	4.85, 4.34, 0.24	1.12, 0.36, 0.05	1.01, 0.12, 0.05
All five areas	4.66, 4.18, 0.23	3.81, 3.41, 0.19	2.95, 2.44, 0.15	1.57, 0.96, 0.08	1.19, 0.47, 0.06	1.01, 0.11, 0.05	1.00, 0.00, 0.05
Whole image	14.22, 13.36, 0.71	9.05, 8.49, 0.45	5.06, 4.66, 0.25	1.55, 0.93, 0.07	1.05, 0.24, 0.05	1.00, 0.00, 0.05	1.00, 0.00, 0.05

Table 8
ARL, SDRL, ARL/20 values of the control chart under different mean shift sizes and in different areas of the image, with the 30 × 30 grids size and in the spatial domain with windowing.

Shifted area	Shift size 0.2	0.5	0.9	2	3	5	10
First area	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC
Second area	Same as IC	Same as IC	Same as IC	15.83, 15.87, 0.79	12.04, 11.67, 0.6	5.08, 4.73, 0.25	4.16, 3.74, 0.21
Third area	16.07, 16.23, 0.8	12.27, 12.47, 0.61	7.48, 6.90, 0.37	2.19, 1.64, 0.11	2.43, 1.85, 0.12	1.64, 1.03, 0.08	1.00, 0.00, 0.05
Fourth area	14.45, 13.66, 0.73	9.99, 9.49, 0.5	5.46, 5.15, 0.27	1.62, 1.00, 0.08	1.62, 1.00, 0.08	1.51, 0.85, 0.07	1.00, 0.00, 0.05
Fifth area	11.85, 11.60, 0.59	7.66, 7.24, 0.38	4.23, 3.70, 0.21	1.41, 0.74, 0.07	1.45, 0.80, 0.07	2.04, 1.43, 0.1	1.00, 0.00, 0.05
All five areas	8.51, 7.83, 0.42	4.92, 4.39, 0.25	2.46, 1.84, 0.12	1.06, 0.26, 0.05	1.09, 0.33, 0.05	1.09, 0.32, 0.05	1.00, 0.00, 0.05
Whole image	11.89, 11.69, 0.59	5.94, 5.67, 0.3	3.03, 2.45, 0.15	1.16, 1.16, 0.06	1.00, 0.05, 0.05	1.00, 0.00, 0.05	1.00, 0.00, 0.05

Table 9
ARL, SDRL, ARL/20 values of the control chart under different mean shift sizes and in different areas of the image, with the 100 × 100 grids size and in the frequency domain.

Shifted area	Shift size 0.2	0.5	0.9	2	3	5	10
First area	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC
Second area	9.89, 9.25, 0.5	10.81, 10.26, 0.54	11.79, 11.43, 0.58	7.57, 6.83, 0.38	4.14, 3.62, 0.21	7.63, 7.12, 0.38	11.23, 11.05, 0.56
Third area	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC
Fourth area	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC
Fifth area	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC
All five areas	9.58, 8.99, 0.48	10.47, 10.83, 0.52	11.74, 11.41, 0.59	7.49, 6.77, 0.37	4.15, 3.76, 0.21	7.64, 7.26, 0.38	10.72, 9.86, 0.54
Whole image	12.26, 11.60, 0.61	6.61, 6.04, 0.33	3.38, 2.86, 0.17	1.20, 0.47, 0.06	1.01, 0.09, 0.05	1.00, 0.00, 0.05	1.00, 0.00, 0.05

Table 10
ARL, SDRL, ARL/20 values of the control chart under different mean shift sizes and in different areas of the image, with the 50 × 50 grids size and in the frequency domain.

Shifted area	Shift size 0.2	0.5	0.9	2	3	5	10
First area	17.59, 17.63, 0.87	17.60, 16.73, 0.88	18.72, 18.64, 0.94	18.60, 18.37, 0.93	17.05, 16.32, 0.85	11.53, 11.05, 0.58	19.93, 19.81, 0.99
Second area	19.72, 19.63, 0.99	19.68, 19.99, 0.99	17.70, 17.14, 0.88	8.42, 7.99, 0.42	4.53, 3.85, 0.23	7.53, 6.71, 0.38	19.86, 19.08, 0.99
Third area	10.02, 9.86, 0.5	6.50, 6.05, 0.32	3.50, 2.89, 0.17	1.30, 0.60, 0.06	1.03, 0.19, 0.05	1.71, 1.09, 0.08	4.92, 4.38, 0.25
Fourth area	13.76, 13.35, 0.69	9.86, 9.15, 0.49	5.35, 4.94, 0.27	1.68, 1.10, 0.08	1.61, 0.96, 0.08	12.55, 11.75, 0.63	1.74, 1.18, 0.09
Fifth area	12.61, 11.87, 0.63	8.51, 8.21, 0.43	4.54, 3.91, 0.23	1.47, 0.82, 0.07	1.21, 0.50, 0.06	6.37, 5.89, 0.32	2.55, 2.02, 0.13
All five areas	6.58, 6.20, 0.33	3.81, 3.44, 0.19	2.04, 1.45, 0.1	1.03, 0.19, 0.05	1.00, 0.02, 0.05	1.50, 0.85, 0.07	1.31, 0.64, 0.31
Whole image	12.19, 11.65, 0.61	5.87, 5.27, 0.3	3.06, 2.50, 0.15	1.18, 0.46, 0.06	1.00, 0.07, 0.05	1.00, 0.00, 0.05	1.00, 0.00, 0.05

uniformly. It is worth mentioning again that their chart was only analyzed in the frequency domain, and unlike our memory-less scheme, it is a memory-type chart, which should inherently be faster in detecting small shifts.

Next, for this gridding case, we first apply the windowing to each grid and then feed them into the CNN. For this sub-scenario, after training the CNN, the validation accuracy is computed as 96.25 % and the test accuracy is computed as 93.58 %. The *UCL* in the case is estimated as 0.83. We computed the empirical false alarm rate for this case as 0.0502, which is again very close to its theoretical value.

As it can be seen in Table 4, windowing makes the control chart perform a bit better in the spatial domain as well, when this gridding case is used. Shifts in the second area (the one in the middle of the image and shared equally by four grids), as well as simultaneous shifts involving that area, are detected well under medium and large shift sizes. The whole image shifts detections are also improved when windowing is used.

Case 2: 50 × 50 grids.

The speckle pattern in this case is divided into equal-sized grids of size 50 × 50 (which makes 144 grids in the image), as shown in Fig. 8. For when the windowing is not applied to the grids and they are directly inputted, after training the CNN the validation accuracy is computed as 99.64 % and the test accuracy is computed as 99.69 %. The *UCL* in the case is estimated as 0.721, and the empirical false alarm rate as 0.0486.

The result of the control chart performance evaluation in this case is presented in Table 5. Similar to the previous case, the control chart is unable to detect localized shifts and its performance is mostly the same as the IC performance. However, again similar to the previous case, it performs well in detecting shifts when the whole image is shifted uniformly; with a little performance improvement in this gridding case.

In addition, when we compare this chart to the chart designed by Sabahno and Khodadad (2025), under the same gridding (its Table 4), their chart performs better overall in detecting localized shifts, than our chart under this case-scenario performs better in detecting small whole image shifts.

Next, for when the windowing is applied to the grids before feeding them into the CNN for training, the validation accuracy is computed as 97.13 % and the test accuracy is computed as 94.54. The *UCL* in the case is estimated as 0.802, and the empirical false alarm rate as 0.0479.

As in Table 6, the windowing in this gridding case shows massive improvements, especially in detecting localized and simultaneous shifts. Except for the second area, other areas are detected very rapidly, especially larger areas (areas 4 and 5) when windowing is applied. Moreover, when all the areas shift together, the chart's performance in this case-scenario is always much better than in a case in which any of the local areas is individually shifted, which shows the chart's absolute power in detecting simultaneous shifts in this case.

In addition, Unlike the previous sub-scenario of this gridding case, with windowing our scheme performs better than that of Sabahno and Khodadad (2025), even in localized and simultaneous shifts detections; except for the second area.

Case 3: 30 × 30 grids.

Table 11
ARL, SDRL, ARL/20 values of the control chart under different mean shift sizes and in different areas of the image, with the 30 × 30 grids size and in the frequency domain.

Shifted area	Shift size 0.2	0.5	0.9	2	3	5	10
First area	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC	Same as IC
Second area	14.78, 13.77, 0.74	11.60, 11.17, 0.58	7.42, 6.99, 0.37	2.27, 1.66, 0.11	1.55, 0.87, 0.08	6.29, 5.95, 0.31	18.98, 18.67, 0.95
Third area	15.91, 15.13, 0.8	11.78, 11.60, 0.6	6.75, 6.25, 0.34	1.65, 1.04, 0.08	1.05, 0.22, 0.05	1.59, 0.95, 0.08	2.21, 1.65, 0.11
Fourth area	10.06, 9.72, 0.5	6.84, 6.25, 0.34	3.84, 3.41, 0.19	1.39, 0.73, 0.07	1.19, 0.48, 0.06	3.00, 2.42, 0.15	1.29, 0.63, 0.06
Fifth area	13.17, 12.98, 0.66	8.58, 7.97, 0.43	5.00, 4.36, 0.25	1.39, 0.73, 0.07	1.10, 0.33, 0.05	5.19, 4.63, 0.26	1.40, 0.74, 0.07
All five areas	6.63, 6.03, 0.33	3.94, 3.33, 0.2	2.05, 1.53, 0.1	1.02, 0.16, 0.05	1.00, 0.03, 0.05	1.25, 0.57, 0.06	1.03, 0.18, 0.05
Whole image	13.21, 12.71, 0.66	7.07, 6.71, 0.35	3.39, 2.83, 0.17	1.22, 0.52, 0.06	1.01, 0.11, 0.05	1.00, 0.00, 0.05	1.00, 0.00, 0.05

The speckle pattern in this case is divided into equal-sized grids of size 30 × 30 (which makes 400 grids in the image), as shown in Fig. 9. When windowing is not applied, after training the CNN the validation accuracy is computed as 99.66 % and the test accuracy is computed as 99.64 %. The *UCL* in the case is estimated as 0.99, and the empirical false alarm rate as 0.0527.

Table 7 contains the chart's performance in this gridding case when windowing is not applied. Based on this table, localized shifts are now detected much faster when this small gridding case is utilized. However, some performance fluctuations are apparent. Normally in control charts, as the shift size increases, the chart's speed in detecting shifts is supposed to increase as well (i.e., the ARL values decrease). However, in this case-scenario this general rule only applies when considering medium to large shifts (2–10) and also when the shifted areas are the first and fourth areas (left side of the speckle pattern). Moreover, in here again when all the areas shift together, the chart's performance is always much better than in a case in which any of the local areas is individually shifted. In addition, in this case as well the chart performs well in detecting shifts when the whole image shifts uniformly. However, it performs a little worse compared to the previous cases.

In addition, when we compare this chart to the chart designed by Sabahno and Khodadad (2025), under the same gridding (its Table 5), the proposed chart performs much better in detecting localized shifts (except for medium shift sizes in the second area and medium to large shift sizes in the third area), but their chart performs better in detecting shifts when the whole image is shifted. Moreover, the proposed chart is much faster in detecting simultaneous shifts than their chart.

When windowing is applied in this gridding case, after training the CNN, the validation accuracy is computed as 94.87 % and the test accuracy is computed as 94.46 %. The *UCL* in the case is estimated as 0.966, and the empirical false alarm rate for this case as 0.0496.

Table 8 presents the results of this analysis. Comparing this analysis to that of Table 7 shows that, except for the frist area whose shifts can only be detected without windowing, with windowing the chart performs better in detecting medium to large localized shifts sizes. So there is no clear winner as to which one is better at detecting localized shifts in this gridding case. In addition, with windowing the chart performs better in detecting the whole image shifts, even when it is compared to the chart of Sabahno and Khodadad (2025). Also, unlike the previous sub-scenario, there is no performance fluctuation at all when windowing is applied.

4.1.2. *Second Scenario: Speckle image in the frequency domain*

In this scenario, first, the image in Fig. 4 is divided into equal-sized grids, then windowing, Fourier transformation, and high pass filtering are applied to each grid individually (as described in Section 2), and finally, they are fed into the CNN for training. So, in this scenario $F'_1(u, v)$ (Equation 5) and $F'_2(u, v)$ (Equation 6) are fed into the CNN for training.

This scenario's objective is to assess the model's effectiveness after transforming grids into the frequency domain and applying high-pass filtering to emphasize fine texture shifts.

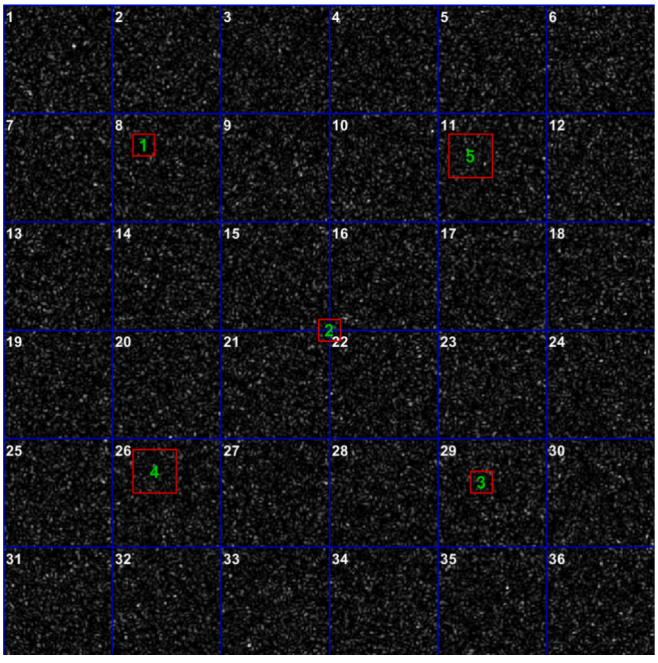

Fig. 7. The shifted areas in the speckle pattern, when the grids size is 100 × 100.

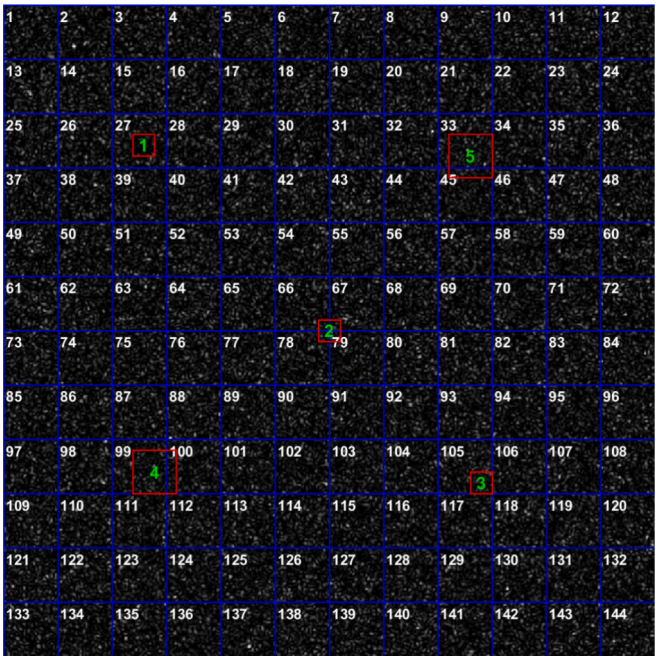

Fig. 8. The shifted areas in the speckle pattern, when the grids size is 50 × 50.

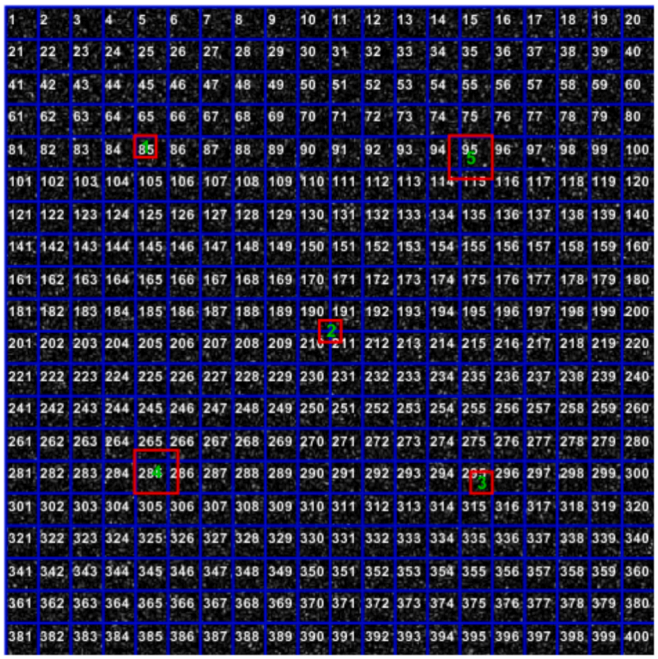

Fig. 9. The shifted areas in the speckle pattern, when the grids size is 30 × 30.

We apply FFT to each Blackman-windowed grid and retain the magnitude spectrum. High-pass emphasis filters are used to suppress background trends. The same CNN architecture is trained on frequency domain data. Only the input layer is differnet compared to the previous scenario. Here, the grids are inputted after transformation into the frequesncy domain. The method for estimating the *UCLs* and ARLs are also the same. Tables 9-11 show the ARL performance for various grid sizes and shift magnitudes.

Case 1: 100 × 100 grids.

For this case-scenario, after training the CNN, the validation accuracy is computed as 94.17 % and the test accuracy is computed as 92.78. The *UCL* in the case is estimated as 0.693. We computed the empirical false alarm rate for this case as 0.052, which is very close to its theoretical value of 0.05.

Based on Table 9, when the speckle pattern is analyzed in the frequency domain, similar to the best-performing sub-scenario of the previous scenario (with windowing), under the same gridding (100 × 100 grids), the chart is only able to detect shifts in the second area (and simultaneous shifts involving that area). However, small localized shifts sizes are better detected in this scenario (frequency domain), although the chart's performance in this scenario is not consistent with how a control chart should normally behave when the shift size increases/decreases and performance fluctuation is visible at points. Regarding the whole image shifts, this scenario only performs better in detecting very small shift sizes compared to the best of the previous scenario.

Moreover, comparing the proposed chart's performance under the second scenario to the chart in Sabahno and Khodadad (2025), under similar gridding (its Table 3), reveals that both charts perform almost the same in detecting localized shifts, except for the second area shifts which are much better detected by the proposed control chart. Also, our chart performs faster in detecting small shifts when the whole image is shifted uniformly.

Case 2: 50 × 50 grids.

For this case-scenario, after training the CNN, the validation accuracy is computed as 93.65 % and the test accuracy is computed as 93.99 %. The *UCL* in the case is estimated as 0.912, and the empirical false alarm rate as 0.0511.

Table 10 shows that, compared to the best sub-scenario of the previous scenario which is still the with-windowing one, there is no clear winner. While some shifts in the second area can only be detected when analyzed in the frequency domain when this gridding is used, shifts in the larger areas (4 and 5) are better detected when the spatial domain with windowing is used. Regarding the shifts in the first area, only under a very small shift size the chart performs better in the frequency domain, while regarding the shifts in the third area, the current scenario is clearly a winner, except for the very large shift size (10). Regarding simultaneous and whole image shifts (the last two rows of the tables), the best chart in the previous scenario (with windowing) performs a little bit better than the current scenario chart under this gridding, although their performance is rather the same.

Furthermore, compared to the previous case of this scenario, localized and simultaneous shifts are detected much faster in this gridding case, except for shifts in the second area which are detected faster under the previous gridding case.

In addition, comparing our proposed chart under this scenario and gridding case to Sabahno and Khodadad (2025) chart (its Table 4) reveals that the proposed chart is overall better in detecting localized shifts (except for large shifts in the smaller areas; areas 1, 2, and 3) and also small shifts.

Case 3: 30 × 30 grids.

For this case-scenario, after training the CNN, the validation accuracy is computed as 92.51 % and the test accuracy is computed as 91.77 %. The *UCL* in the case is estimated as 0.97, and the empirical false alarm rate as 0.0498.

Table 11 contains the result of this gridding case in this scenario. Regarding localized shifts detection, in some areas, the chart in this gridding case performs better, and in others, the chart in the previous gridding case performs better. The same applies when comparing the localized shifts detection of the same gridding under the first and second scenarios. Moreover, when detecting simultaneous shifts under this gridding case and the previous gridding case of this scenario, the chart's performance is not only very well but almost similar; and in both cases it is much better than detecting any localized shift individually. In addition, when the whole image is shifted uniformly, the chart in the previous gridding case performs better than the current gridding case (i.e., 50 × 50 case better than 30 × 30 case).

Furthermore, comparing this table's results to the previous scenario gives us different outcomes. Regarding the first area, the chart designed in the spatial domain without windowing is the only one able to detect its shifts. Regarding the second area, the current chart performs much better, except for very large shift size detection which the spatial domain charts perform much better. The shifts in the third area are faster detected under the current scenario. The shifts in the fourth and fifth areas (larger areas), and also simultaneous shifts are detected better when the analysis is performed in the spatial domain. The whole image shifts are detected better when the analysis is performed in the spatial domain with windowing.

Finally, comparing the proposed chart in this gridding case to the chart of Sabahno and Khodadad (2025) (its Table 5) reveals that while localized and simultaneous shifts are detected much faster by the proposed chart, their chart is a bit faster in detecting the whole image shifts.

4.2. Online monitoring and localization

In this section, we demonstrate how the proposed methodology can be implemented for real-time process monitoring. Since the implementation is similar regardless of the domain (scenario) and also the grids size, we select the third gridding case of the second scenario for this purpose (frequency domain with 30 × 30 grids). We also select a few scenarios for implementing artificial shifts. In each scenario, we take consecutive samples and continue process monitoring until 10 samples have been taken, regardless of how many signals are received before then.

First, we only shift the third area by **2**: i.e., N(**2**,**I**). Note that, as in our simulation analyses, when any of the areas is shifted, the other parts of

Table 12
Online process monitoring of consecutive samples, with the grids size 30 × 30 and the mean shift size **2**, when the third area shifts.

i	MPr_i	UCL	Process Status	Identified Grids Numbers	Identified Grids Pr_i Values
1	1	0.97	Out-of-Control	295	1
2	0.995	0.97	In-Control	–	–
3	0.96	0.97	In-Control	–	–
4	0.96	0.97	In-Control	–	–
5	0.84	0.97	In-Control	–	–
6	0.999	0.97	Out-of-Control	295	0.999
7	0.977	0.97	Out-of-Control	295	0.977
8	0.80	0.97	In-Control	–	–
9	0.95	0.97	In-Control	–	–
10	0.96	0.97	In-Control	–	–

the image are uniformly shifted by N(**0**,**I**) to represent normal fluctuations. The result of this shifting scenario is outlined in Table 12. According to this table, the chart signals three times during the first ten samplings, and in all cases correctly identifies the shifted grid's number as 295. Fig. 9 shows that the third area is completely located inside grid 295. So, the actual affected grid in this case is grid 295.

Next, we similarly shift the fourth area (with mean 2). Table 13 shows that, as before, the chart signals three times during the first ten consecutive samplings. However, the identified grids are not the same each time. Fig. 9 shows that the fourth area completely covers grid 285 and the second share belongs to grid 265, and then grids 286 and 266, respectively. So, the actual affected grids in this case are grids 265, 266, 285, and 286. However, during the first ten samplings, grid 265 is the one that gets always identified, and at sample 9, 285 is also identified alongside 265.

To see how the localization is done under different shift sizes, we increase the mean shift to **5** in the previous scenario and perform our online monitoring again. According to Table 14, the chart signals four times during the first ten samplings. In two of those signals, grid 265 is identified, and in the other two, grid 285 is identified. In both scenarios of the fourth area shifting (Tables 13 and 14), grids 266 and 286 never get identified and this must be because of their lowest share in the fourth area.

In the next shifting scenario, we shift the third, fourth, and fifth areas simultaneously, each shifted separately using N(**2**,**I**). Fig. 9 shows that the actual affected grids by shifts in these three areas are grids 94, 95, 114, 115, 265, 266, 285, 286, and 295.

As can be seen in Table 15, the chart signals much faster when these areas shift simultaneously, and all the samples show OC during the first ten consecutive samplings. The first sample correctly identifies all the

Table 13
Online process monitoring of consecutive samples, with the grids size 30 × 30 and the mean shift size **2**, when the fourth area shifts.

i	MPr_i	UCL	Process Status	Identified Grids Numbers	Identified Grids Pr_i Values
1	0.94	0.97	In-Control	–	
2	0.95	0.97	In-Control	–	
3	0.83	0.97	In-Control	–	
4	0.93	0.97	In-Control	–	
5	0.88	0.97	In-Control	–	
6	0.997	0.97	Out-of-Control	265	0.997
7	0.94	0.97	In-Control	–	
8	0.98	0.97	Out-of-Control	265	0.98
9	0.995	0.97	Out-of-Control	265, 285	0.995, 0.978
10	0.92	0.97	In-Control	–	

Table 14
Online process monitoring of consecutive samples, with the grids size 30 × 30 and the mean shift size **5**, when the fourth area shifts.

i	MPr_i	UCL	Process Status	Identified Grids Numbers	Identified Grids Pr_i Values
1	0.973	0.97	Out-of-Control	285	0.973
2	0.91	0.97	In-Control	–	–
3	0.93	0.97	In-Control	–	–
4	0.94	0.97	In-Control	–	–
5	0.975	0.97	Out-of-Control	265	0.975
6	0.88	0.97	In-Control	–	–
7	0.93	0.97	In-Control	–	–
8	0.973	0.97	Out-of-Control	285	0.973
9	0.972	0.97	Out-of-Control	265	0.972
10	0.86	0.97	In-Control	–	–

Table 15
Online process monitoring of consecutive samples, with the grids size 30 × 30 and the mean shift size **2**, when the third, fourth, and fifth areas shift.

i	MPr_i	UCL	Process Status	Identified Grids Numbers	Identified Grids Pr_i Values
1	0.997	0.97	Out-of-Control	265, 285, 295, 95	0.985, 0.993, 0.996, 0.997
2	1	0.97	Out-of-Control	95, 295	0.989,1
3	0.999	0.97	Out-of-Control	285, 265, 295	0.984, 0.998, 0.999
4	1	0.97	Out-of-Control	265, 295	0.999,1
5	0.999	0.97	Out-of-Control	95	0.999
6	0.999	0.97	Out-of-Control	95, 295, 265	0.976, 0.981, 0.999
7	1	0.97	Out-of-Control	95	1
8	0.999	0.97	Out-of-Control	265, 295	0.995, 0.999
9	1	0.97	Out-of-Control	95, 265	0.996,1
10	0.996	0.97	Out-of-Control	295	0.996

Table 16
Online process monitoring of consecutive samples, with the grids size 30 × 30 and the mean shift size **5**, when the third, fourth, and fifth areas shift.

i	MPr_i	UCL	Process Status	Identified Grids Numbers	Identified Grids Pr_i Values
1	0.96	0.97	In-Control	–	–
2	0.99	0.97	Out-of-Control	295	0.99
3	0.97	0.97	Out-of-Control	265	0.97
4	1	0.97	Out-of-Control	295	1
5	0.980	0.97	Out-of-Control	285,295	0.978, 0.980
6	1	0.97	Out-of-Control	265,295	0.972,1
7	0.998	0.97	Out-of-Control	95	0.998
8	0.991	0.97	Out-of-Control	265,295	0.981, 0.991
9	0.993	0.97	Out-of-Control	265,295	0.978, 0.993
10	1	0.97	Out-of-Control	95, 265	0.982,1

Table 17
Online process monitoring of consecutive samples, with the grids size 50 × 50 and the mean shift size 3, when the third, fourth, and fifth areas shift.

i	MPr_i	*UCL*	*Process Status*	*Identified Grids Numbers*	*Identified Grids* Pr_i *Values*
1	0.995	0.912	Out-of-Control	105,33	0.995,0.975
2	0.996	0.912	Out-of-Control	105,33	0.996,0.980
3	1	0.912	Out-of-Control	105,33,99	1,0.977,0.931
4	0.999	0.912	Out-of-Control	105,33,99	0.999,0.992,0.973
5	0.999	0.912	Out-of-Control	105,33	0.999,0.986
6	1	0.912	Out-of-Control	105,99	1,0.991
7	0.984	0.912	Out-of-Control	105,33,99	0.984,0.982,0.971
8	0.996	0.912	Out-of-Control	105,99	0.996,0.985
9	1	0.912	Out-of-Control	105,99,33	1,0.980,0.921
10	0.999	0.912	Out-of-Control	105,33,99	0.999,0.988,0.957

main grids involved with these areas, but the other samples identify fewer grids. However, none of them incorrectly identifies any grid as shifted (same as before).

We also apply the shift of N(**5**,**I**) to the previous shifting scenario, and the results are presented in Table 16. The results show fewer OC signals during the first ten consecutive samplings, and also, fewer grids are identified at each sampling compared to the previous case (yet correctly identified).

To better analyze the identification and localization capability of the proposed scheme, we also test the gridding case of 50 × 50 as well for the previous simultaneous shift scenario. We also increase the mean shift size for all three areas to **3**. As it is clear in Fig. 8, all the affected grids by these three areas are grids 33, 45, 99, 100, and 105. However, the main (highly affected) grids involved with these three areas are grids 33, 99, and 105. Table 17 shows that during the first ten consecutive samplings, at least two of them are identified at each sampling, and in half of the samplings, all three are identified and reported. However, grid 105 is always identified, and this must be because the third area is completely positioned inside this grid.

5. Concluding remarks

In this paper, we designed a custom CNN architecture and used it to construct a control chart for detecting localized and subtle speckle shifts. We also carefully designed the training methodology to achieve these goals. Moreover, to facilitate localization and simultaneous shifts detection, instead of using the entire images, we divided them into equal-sized grids and fed these grids into the CNN for training. We considered two-channel inputs: one containing the shifted grids and the other representing the difference between the shifted and reference grids. We divided the input dataset into equal IC and OC datasets, each uniquely generated to improve the performance. The control chart was designed based on the maximum out-of-control probability among all the grids of the image. The way this control scheme was designed not only enabled the detection of localized and simultaneous shifts but also allowed for the identification (localization) of the shifted grids within the image; all in a single step. We estimated the chart's control limit based on that maximum probability to first detect signals, and then considered the grids with their OC probability over that limit, as shifted. For our simulation studies, we considered two different scenarios. In the first scenario, grids were fed into the CNN in their spatial form. In the first scenario, two sub-scenarios were considered as well: with and without windowing. In the second scenario, the analysis was performed in the frequency domain and windowing, Fourier transform, and high-pass filtering were applied to the grids before being fed into the CNN for training. We also examined different grid sizes (small, medium, and large) and evaluated the control chart's performance under various shift sizes and shifted areas. Additionally, through illustrative examples, we demonstrated how the proposed scheme can be used for online monitoring, detecting, and localizing shifted grids.

The simulation results indicated that the CNN-based control chart's performance heavily depends on grid size, shift size, and the location and size of the affected area, with no clear overall winner among the considered scenarios and sub-scenarios. However, each has its strengths. In the spatial domain, the control chart performed better for detecting whole-image shifts, while windowing generally improved the detection of localized and simultaneous shifts, especially in medium-sized grids. In contrast, the control chart in the frequency domain generally performed better for detecting small localized shifts when the grid size was sufficiently large to resolve spatial patterns effectively, yet remained constrained enough to avoid averaging out subtle local variations. Additionally, real-time monitoring effectively detected and localized shifts, though grid identification, especially in simultaneous shift cases, was dependent on the shift size, affected areas, and grid size. Moreover, the proposed CNN-based control chart outperformed existing methods, particularly in detecting small global shifts, localized shifts, and simultaneous shifts.

Several promising directions can further enhance this framework. First, detecting more complex shift types — such as rotational shifts — would broaden the model's applicability beyond translational motion. Second, incorporating adaptive or dynamic grid selection strategies may improve both detection sensitivity and localization accuracy. Third, future works can focus on improving within-grid interpretability. Importantly, validating the framework on experimentally acquired speckle images from real industrial systems (e.g., wear or deformation monitoring) is beneficial for assessing practical viability. Finally, evaluating the method against a broader range of image-based monitoring models and analyzing statistical properties such as multiple testing effects across grids would further strengthen its reliability and benchmarking value.

CRediT authorship contribution statement

Hamed Sabahno: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Davood Khodadad:** Writing – review & editing, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors gratefully acknowledge the financial support of the Kempe Foundation (Kempestiftelserna) (https://www.kempe.com/) for grant number JCSMK22-0144, which made this research possible. The

authors are also grateful for the journal editorial board and anonymous reviewers' comments, which have led to significant improvement of our paper.

Appendix. Fundamentals of convolutional neural networks

CNNs consist of many different layers: Input Layer, Convolutional Layers, Batch Normalization Layers, Activation Layers, Pooling Layers (downsampling), Fully Connected Layers, Dropout Layers, Softmax & Classification Layers.

The input layer ensures that the image data fed into the network matches the dimensions expected by the subsequent layers, allowing the CNN to process and learn from the image data effectively.

The convolutional layer is the core building block of a CNN. The more convolutional layers a CNN has, the deeper it becomes. It applies a series of filters (kernels) to the input images to produce feature maps. Each filter is a small matrix that slides over the image, performing element-wise multiplications and summing the results. This operation is known as convolution:

$$(I^{*}K)(i,j) = \sum_{m=0}^{h-1}\sum_{n=0}^{w-1} I(i+m, j+n)K(m,n), \tag{1}$$

where I is the $H \times W$ input image and K is the $h \times w$ kernel/filter applied to the image. The number of filters determines how many such kernels are used in the layer, (i,j) are the coordinates in the output feature map with i from 0 to H-h and j from 0 to W-w, and (m,n) are the coordinates within the filter K. In addition, in convolution layers, padding is the process of adding extra pixels (usually zeros) around the input image before applying a convolution operation. This helps control the output size and preserve spatial information, especially near the edges. If no padding is used (also called valid convolution in this case), the convolution shrinks the image and edge pixels get used less, leading to loss of detail. The same padding keeps the input and output size the same, while full padding makes the output size larger than the input. Another operation considered in convolution is called dilation. Dilation expands the receptive field without increasing filter size by inserting gaps (zeros) between kernel elements. This helps detect larger patterns while preserving details. If the dilation factor = 1 (standard convolution), there will be no gaps, and it is suitable for a small receptive field. If the dilation factor = 2, there will be one zero between kernel elements and it suits a larger receptive field. If the dilation factor = 3+, there will be more gaps and it covers even larger spatial features.

The batch normalization layer normalizes the activations of the previous layer for each mini-batch. This helps to stabilize and accelerate training. Mathematically, if x is the input to the batch normalization layer, the output $\widehat{x}$ is computed as:

$$\widehat{x} = \frac{x-\mu}{\sqrt{\sigma^2+\varepsilon}}.\gamma + \beta, \tag{2}$$

where μ is the mean of the mini-batch, σ^2 is the variance of the mini-batch, ε is a small constant for numerical stability, and γ and β are learnable scaling and shifting parameters.

The activation function introduces non-linearity into the network, enabling it to learn more complex patterns. The Rectified Linear Unit (ReLU) function is commonly used, which is defined as:

$$ReLU(x) = \begin{cases} x & if x > 0 \\ 0 & if x \le 0 \end{cases}, \tag{3}$$

The pooling layer reduces the spatial dimensions (height and width) of the feature maps while retaining the most important information. Max pooling is a common pooling technique where the maximum value is taken from each patch of the feature map.

The dropout layer is a crucial component in neural network architectures which can be used to prevent overfitting by randomly dropping units during training. The output $\widetilde{h}_i$ of neuron i after applying dropout is given by:

$$\widetilde{h}_i = \begin{cases} h_i/(1-p) & if\ r_i = 1 \\ 0 & if\ r_i = 0 \end{cases}, \tag{4}$$

where r_i is the dropout mask indicating whether the neuron is dropped or not. Note that the kept neurons are then scaled by $1/(1-p)$ to compensate for the fact that dropout reduces the number of active neurons.

The fully connected layers connect each neuron in the current layer to every neuron in the previous layer. They are typically used at the end of the network to produce the final output. For a fully connected layer with input vector x and weight matrix W, the output vector y is computed as:

$$y = W.x + b, \tag{5}$$

where b is the bias vector.

The softmax Layer converts the raw output scores (logits) of the network into probabilities that sum up to 1, which represent the likelihood of each class. Given the input vector $\boldsymbol{z} = [z_1, z_2, \cdots, z_{tc}]$, where tc is the number of classes, the softmax function for class i is applied as:

$$p_i = \frac{e^{z_i}}{\sum_{j=1}^{tc} e^{z_j}}, \tag{6}$$

where p_i is the probability of class i and z_i is the raw score (logit) for class i. The output in this layer is a vector where each element in it represents the probability of a class, i.e., $\boldsymbol{p} = [p_1, p_2, \cdots, p_{tc}]$.

The purpose of *the classification Layer* is to compute the classification loss and to evaluate the performance of the network during training and validation. It also handles the final step of converting the network's output into a predicted class label. It contains two components:

i) Loss Function: For a multi-class classification problem, the classification Layer typically uses the cross-entropy loss function. For true labels $\boldsymbol{y} = (y_1, y_2, \cdots, y_{tc})$, where y_c is 1 if class c is the true class, and 0 otherwise, and predicted labels $\widehat{\boldsymbol{y}} = (\widehat{y}_1, \widehat{y}_2, \cdots, \widehat{y}_{tc})$, where $\widehat{y}_c$ is the predicted probability for class c, the cross-entropy loss is calculated as $L(\widehat{\boldsymbol{y}}, \boldsymbol{y}) = -\sum_{c=1}^{tc} y_c log(\widehat{y}_c)$.
ii) Output: The classification Layer outputs the loss value for a batch of data, which is used to update the network's weights during training. It also determines the predicted class by selecting the class with the highest probability from the softmax Layer.

The Softmax and Classification layers are typically used together at the end of a neural network to handle the final stage of classification. Here is how these layers work together:

Forward Pass:

- The network's final fully connected (dense) layer produces raw scores (called logits) for each class.
- The Softmax layer converts these logits into a probability distribution over the classes.
- The Classification layer (or loss layer) computes the loss (e.g., cross-entropy loss) based on the true class labels and the predicted probabilities from the Softmax layer.

Backward Pass (Training):

- During backpropagation, the Classification layer computes the gradient of the loss function with respect to the network's output.
- These gradients are propagated backward through the Softmax and earlier layers to compute gradients with respect to the weights.
- Finally, the network's weights are updated (e.g., via gradient descent) to minimize the loss.

Other than layers, which define the main structure of a CNN, another crucial factor in designing a CNN model is the choice of hyperparameters, and their values/types should be carefully selected for any specific need.

The main hyperparameters in a CNN architecture are learning rate, mini-batch size, number of epochs, number of filters (kernels), dropout rate, weight decay, and optimizer type.

The learning rate (η) controls the step size at each iteration while moving toward a minimum of the loss function (see optimizer types below for details; Equations 10 and 11).

The mini-batch size (B) is the number of training examples used to calculate the gradient at each iteration. The batch gradient descent is**:**

$$\nabla_\theta J(\theta) = \frac{1}{B}\sum_{i=1}^{B} \nabla_\theta J(\theta; x_i, y_i), \tag{7}$$

where θ represents the parameters of the model (weights and biases), B is the mini-batch size, and (x_i, y_i) represents the training samples (x_i) and their corresponding labels (y_i).

The number of epochs is the number of complete passes through the entire training dataset. Each epoch involves one full cycle of updating the weights with all the training samples.

The number of iterations in each Epoch is calculated as $\frac{NumberofTrainingSamples}{B}$.

Number of filters determines the number of features detected in each layer. Filters (or kernels) are used in convolutional layers to detect features in the input image.

Dropout is a regularization technique to prevent overfitting by randomly setting a fraction p of the input units to 0 at each update during training.

$$\widetilde{h} = \frac{h}{1-p}.r, \tag{8}$$

where h is the input to the dropout layer (the activations from the previous layer), p is the dropout rate, r is a random binary mask where each element is 1 with probability $1 - p$ and 0 with probability p, and $\widetilde{h}$ is the output after applying dropout.

Weight decay (λ) is a regularization technique that adds a penalty proportional to the magnitude of the weights to the loss function. The regularization term is:

$$J_{total} = J(\theta) + \lambda\|\theta\|^2, \tag{9}$$

where $J(\theta)$ is the original loss function and $\lambda\|\theta\|^2$, with $\|\theta\|^2 = \sum_i \theta_i^2$ (sum of squared weights), is the weight decay term.

Optimizers play a crucial role in the efficiency and effectiveness of the training process, addressing various challenges in gradient-based optimization. Common optimizers include Stochastic Gradient Descent (SGD) and Adaptive Moment Estimation (Adam). Adam is built on concepts from two other optimization methods: Momentum and RMSProp (Root Mean Square Propagation), combining them in a way that adapts the learning rate for each parameter. Below, the mechanisms of SGD and Adam are explained.

Stochastic Gradient Descent (SGD): In SGD, the parameters θ are updated at each iteration t according to the following rule:

$$\theta_{t+1} = \theta_t - \eta\nabla_\theta J(\theta_t), \tag{10}$$

where t is the iteration step, η is the learning rate, and $\nabla_\theta J(\theta_t)$ is the gradient of the loss function J with respect to θ at step t. Here, η directly scales the

gradient, controlling the size of the step taken towards the minimum of the loss function.

Adaptive Moment Estimation (Adam): In Adam, the update rule is more complex and involves the estimation of both the first and second moments of the gradients. The updates are as follows

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_\theta J(\theta_t),$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)\nabla_\theta J(\theta_t)^2,$$

$$\widehat{m}_t = \frac{m_t}{1-\beta_1^t},\ \widehat{v}_t = \frac{v_t}{1-\beta_2^t},$$

$$\theta_{t+1} = \theta_t - \eta\frac{\widehat{m}_t}{\sqrt{\widehat{v}_t}+\varepsilon}, \tag{11}$$

where m and v are estimates of the first and second moments of the gradient, β_1 (mean of the gradients, typically set to 0.9) and β_2 (uncentered variance of the gradients, typically set to 0.999) are the hyperparameters for these estimates, and ε is a small constant (typically set to 10^{-8}). η scales the ratio of the bias-corrected first-moment estimate $\widehat{m}_t$ to the square root of the bias-corrected second-moment estimate $\widehat{v}_t$, ensuring that updates are well-scaled for different parameters.

Data availability

Data will be made available on request.

References

Abdel-Nasser, M., Melendez, J., Moreno, A., Omer, O. A., & Puig, D. (2017). Breast tumor classification in ultrasound images using texture analysis and super-resolution methods. *Engineering Applications of Artificial Intelligence, 59*, 84–92. https://doi.org/10.1016/j.engappai.2016.12.019

Amirkhani, F., & Amiri, A. (2020). A novel framework for spatiotemporal monitoring and post-signal diagnosis of processes with image data. *Quality and Reliability Engineering International, 36*, 705–735. https://doi.org/10.1002/qre.2600

Chao, X., Yang, F., Sun, G., & Ding, J. (2024). Measurement of spatial coherence of partially coherent light by spatial averaging of speckle pattern. *Optics & Laser Technology., 176*, Article 110915. https://doi.org/10.1016/j.optlastec.2024.110915

Choi, J. G., Kim, D. C., Chung, M., Lim, S., & Park, H. W. (2024). Multimodal 1D CNN for delamination prediction in CFRP drilling process with industrial robots. *Computers & Industrial Engineering, 190*, Article 110074. https://doi.org/10.1016/j.cie.2024.110074

Y. Guo, C. Wang, S. Han, G. Kosec, Y. Zhou, L. Wang, M. Abdel Wahab, A deep neural network model for parameter identification in deep drawing metal forming process, Journal of Manufacturing Processes, 133(2025) 380–394. DOI: 10.1016/j.jmapro.2024.11.067.

Harris, F. J. (1978). On the use of windows for harmonic analysis with the discrete Fourier transform. *Proceedings of the IEEE, 66*(1978), 51–83. https://doi.org/10.1109/PROC.1978.10837

K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016 770–778. DOI: 10.1109/CVPR.2016.90..

Hecht, E., & Zajac, A. (1974). *Optics* (4th ed.). London: Pearson.

Hosseinifard, S. Z., Abdollahian, M., & Zeephongsekul, P. (2011). Application of artificial neural networks in linear profile monitoring. *Expert Systems with Applications, 38*, 4920–4928. https://doi.org/10.1016/j.eswa.2010.09.160

Iwase, T., Yamamoto, K., Ra, E., Murotani, K., Matsui, S., & Terasaki, H. (2015). Diurnal variations in blood flow at optic nerve head and choroid in healthy eyes: diurnal variations in blood flow. *Medicine, 94*, e519.

Jamali, R., Babaei-Ghazvini, A., Nazari, E., Panahi, M., Shahabi-Ghahfarrokhi, I., & Moradi, A.-R. (2023). Surface characterization of biodegradable nanocomposites by dynamic speckle analysis. *Applied Surface Science Advances, 16*, Article 100429. https://doi.org/10.1016/j.apsadv.2023.100429

D. Khodadad, Multiplexed Digital Holography incorporating Speckle Correlation (PhD dissertation, Luleå tekniska universitet), 2016. https://urn.kb.se/resolve?urn=urn:nbn:se:ltu:diva-26496.

Khodadad, D., & Sabahno, H. (2025). White-light biospeckle displacement analysis for the assessment of fruit quality. *Optics Letters, 50*, 1277–1280. https://doi.org/10.1364/OL.549334

Khodadad, D., Tayebi, B., Saremi, A., & Paul, S. (2023). Temperature sensing in space and transparent media: Advancements in off-axis digital holography and the temperature coefficient of refractive index. *Applied Sciences, 13*, 8423. https://doi.org/10.3390/app13148423

Khodadadi, Z., Owlia, M. S., Amiri, A., & Fallahnezhad, M. S. (2024). Development of control charts to monitor image data using the contourlet transform method. *Quality and Reliability Engineering International, 40*, 876–898. https://doi.org/10.1002/qre.3441

A.C. Kocabıçak, C. Wang, S. Han, H. Nguyen-Xuan, G. Kosec, L. Wang, M. Abdel Wahab, A deep neural network approach to predict dimensional accuracy of thin-walled tubes in backward flow forming plasticity process, Journal of Manufacturing Processes, 141(2025) 59–80. DOI: 10.1016/j.jmapro.2025.02.044.

Koosha, M., Noorossana, R., & Ahmadi, O. (2022). Two-dimensional wavelet based statistical monitoring of image data. *Quality and Reliability Engineering International, 38*, 3797–3815. https://doi.org/10.1002/qre.3174

Koosha, M., Noorossana, R., & Megahed, F. (2017). Statistical process monitoring via image data using wavelets. *Quality and Reliability Engineering International, 33*, 2059–2073. https://doi.org/10.1002/qre.2167

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet Classification with Deep Convolutional Neural Networks. *Communications of the ACM, 60*, 1097–1105. https://doi.org/10.1145/3065386

Kwon, T. H., Park, J., Jeong, H., & Park, K. (2023). Assessment of Speckle-Pattern Quality using Deep-Learning-based CNN. *Experimental Mechanics, 63*, 163–176. https://doi.org/10.1007/s11340-022-00906-x

K.A. Luong, M. Abdel Wahab, J.H. Lee, Simultaneous imposition of initial and boundary conditions via decoupled physics-informed neural networks for solving initial-boundary value problems, Applied Mathematics and Mechanics, 46(2025) 763–780. DOI: 10.1007/s10483-025-3240-7 .

Megahed, F. M., Wells, L. J., Camelio, J. A., & Woodall, W. H. (2012). A spatiotemporal method for the monitoring of image data. *Quality and Reliability Engineering International, 28*, 967–980. https://doi.org/10.1002/qre.1287

Megahed, F. M., Woodall, W. H., & Camelio, J. A. (2011). A review and perspective on control charting with image data. *Journal of Quality Technology, 43*, 83–98. https://doi.org/10.1080/00224065.2011.11917848

Mohammadzadeh, M., Yeganeh, A., & Shadman, A. (2021). Monitoring logistic profiles using variable sample interval approach. *Computers & Industrial Engineering, 158*, Article 107438. https://doi.org/10.1016/j.cie.2021.107438

Montresor, S., Tahon, M., & Picart, P. (2022). Deep learning speckle de-noising algorithms for coherent metrology: A review and a phase-shifted iterative scheme. *Journal of the Optical Society of America A, 39*, 62–78. https://doi.org/10.1364/JOSAA.444951

D.Q. Nguyen, K.Q. Tran, T.D. Le, M. Abdel Wahab, H. Nguyen-Xuan, A data-driven uncertainty quantification framework in probabilistic bio-inspired porous materials (Material-UQ): An investigation for RotTMPS plates, Computer Methods in Applied Mechanics and Engineering, 435(2025), 117603. DOI: 10.1016/j.cma.2024.117603.

Niaki, S. T. A., & Abbasi, B. (2008). Detection and classification mean-shifts in multiattribute processes by artificial neural networks. *International Journal of Production Research, 46*, 2945–2963. https://doi.org/10.1080/00207540601039809

Nguyen, H., Tran, T., Wang, Y., & Wang, Z. (2021). Three-dimensional shape reconstruction from single-shot speckle image using deep convolutional neural networks. *Optics and Lasers in Engineering, 143*, Article 106639. https://doi.org/10.1016/j.optlaseng.2021.106639

Nguyen-Ngoc, L., Nguyen-Huu, Q., De Roeck, G., Bui-Tien, T., & Abdel-Wahab, M. (2024). Deep neural network and evolved optimization algorithm for damage assessment in a truss bridge. *Mathematics, 12*, 2300. https://doi.org/10.3390/math12152300

Pang, Y., Chen, B. K., Liu, W., Yu, S. F., & Lingamanaik, S. N. (2020). Development of a non-contact and non-destructive laser speckle imaging system for remote sensing of anisotropic deformation around fastener holes. *NDT & E International, 111*, Article 102219. https://doi.org/10.1016/j.ndteint.2020.102219

Qureshi, M. M., Allam, N., Im, J., Kwon, H.-S., Chung, E., & Vitkin, I. A. (2023). Advances in laser speckle imaging: From qualitative to quantitative hemodynamic assessment. *Journal of Biophotonics, 17*, Article e202300126. https://doi.org/10.1002/jbio.202300126

Raffel, M., Willert, C. E., Wereley, S. T., & Kompenhans, J. (2007). *Particle image Velocimetry: A Practical Guide* (2th ed.). New York: Springer.

Sabahno, H., & Amiri, A. (2023). New statistical and machine learning based control charts with variable parameters for monitoring generalized linear model profiles. *Computers & Industrial Engineering, 184*, Article 109562. https://doi.org/10.1016/j.cie.2023.109562

Sabahno, H., & Eriksson, M. (2024). Variable parameters memory-type control charts for simultaneous monitoring of the mean and variability of multivariate multiple linear regression profiles. *Scientific Reports, 14*, 9288. https://doi.org/10.1038/s41598-024-59549-8

Sabahno, H., & Khodadad, D. (2025). A spatiotemporal scheme for process control using image analysis of speckle patterns. *Quality and Reliability Engineering International*. https://doi.org/10.1002/qre.3758

Sabahno, H., & Niaki, S. T. A. (2023). New machine-learning control charts for simultaneous monitoring of multivariate normal process parameters with detection and identification. *Mathematics, 11*, 3566. https://doi.org/10.3390/math11163566

K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint, arXiv:1409.1556 (2015). DOI: 10.48550/arXiv.1409.1556..

Sirohi, R. S. (2020). *Speckle metrology* (1st ed.). Florida: CRC Press.

Shen, H., Wei, B., Ma, Y., & Gu, X. (2023). Unsupervised industrial image ensemble anomaly detection based on object pseudo-anomaly generation and normal image feature combination enhancement. *Computers & Industrial Engineering, 182*, Article 109337. https://doi.org/10.1016/j.cie.2023.109337

Smolovich, A. M., Frolov, A. V., Klebanov, L. D., Laktaev, I. D., Orlov, A. P., Smolovich, P. A., & Butov, O. V. (2024). Identification of a replicable optical security element using laser speckle. *Optics & Laser Technology, 175*, Article 110725. https://doi.org/10.1016/j.optlastec.2024.110725

Torre, I. M. D. L., Montes, M. D. S. H., Flores-Moreno, J. M., & Santoyo, F. M. (2016). Laser speckle based digital optical methods in structural mechanics: A review. *Optics and Lasers in Engineering, 87*, 32–58. https://doi.org/10.1016/j.optlaseng.2016.02.008

Yeganeh, A., Johannssen, A., & Chukhrova, N. (2024). The partitioning ensemble control chart for on-line monitoring of high-dimensional image-based quality characteristics. *Engineering Applications of Artificial Intelligence, 127*, Article 107282. https://doi.org/10.1016/j.engappai.2023.107282

Xu, X., Li, X., Ming, W., & Chen, M. (2022). A novel multi-scale CNN and attention mechanism method with multi-sensor signal for remaining useful life prediction. *Computers & Industrial Engineering, 169*, Article 108204. https://doi.org/10.1016/j.cie.2022.108204

Zdunek, A., Adamiak, A., Pieczywek, P. M., & Kurenda, A. (2014). The biospeckle method for the investigation of agricultural crops: A review. *Optics and Lasers in Engineering, 52*, 276–285. https://doi.org/10.1016/j.optlaseng.2013.06.017

Zhang, X., Chi, Y., Yu, L., Wang, Q., & Pan, B. (2024). Laser speckle DIC revisited: An improved calculation scheme for large deformation measurement. *Optics & Laser Technology, 168*, Article 109913. https://doi.org/10.1016/j.optlastec.2023.109913

Zhou, M., Zhang, Y., Wang, P., Li, R., Peng, T., Min, J., Yan, S., & Yao, B. (2024). Speckle-correlation-based non-line-of-sight imaging under white-light illumination. *Optics & Laser Technology, 170*, Article 110231. https://doi.org/10.1016/j.optlastec.2023.110231