2025 5th International Conference on Artificial Intelligence and Industrial Technology Applications (AIITA)

# An Improved Icon Matching Method for GUI Testing of Mobile Devices

1st Kang Xi*
School of Computer Science and Technology,
Xi'an University of Posts and Telecommunications
Shaanxi Key Laboratory of Network Data Analysis and Intelligent Processing
Xi'an, 710121, China
xikang88608@163.com

2nd Zhongmin Wang
School of Computer Science and Technology,
Xi'an University of Posts and Telecommunications
Shaanxi Key Laboratory of Network Data Analysis and Intelligent Processing
Xi'an, 710121, China
zmwang@xupt.edu.cn

3rd Xiaomin Jin
School of Computer Science and Technology,
Xi'an University of Posts and Telecommunications
Shaanxi Key Laboratory of Network Data Analysis and Intelligent Processing
Xi'an, 710121, China
xmjin@xupt.edu.cn

4th Yanping Chen
School of Computer Science and Technology,
Xi'an University of Posts and Telecommunications
Shaanxi Key Laboratory of Network Data Analysis and Intelligent Processing
Xi'an, 710121, China
ypchen@xupt.edu.cn

***Abstract*—In recent years, with the emergence of Industry 4.0, automated testing technology based on machine vision has been widely used in intelligent manufacturing production lines. However, in the visualization test task of mobile devices, due to the interference of the complex environment of the factory and the problem of the feature images of the GUI controls not being obvious, As a result, when industrial robots perform matching and positioning operations on GUI controls on mobile devices, matching failures and mismatching occur. Firstly, this paper proposes a feature extraction method that combines edge detection algorithm, sift algorithm, and DBSCAN clustering algorithm to extract the feature information of all GUI accurately controls on the mobile device's screen to be detected for further image matching. Secondly, an improved twin neural network is proposed to solve the problem of recognizing template icons and icons to be tested. Based on the original model, the GeLU activation function is introduced by replacing the backbone structure of the RepVGG network and adding an attention mechanism to the feature extraction part. The method was evaluated on a specially constructed GUI control icon dataset, and the accuracy of the improved network model reached 98.6%. When performing the GUI control matching task, the total elapsed time for matching is much shorter than other existing algorithms, and the correct rate of matching to the correct icon in the batch test also reaches 94.4%. Applying the proposed matching method to the visualization testing task effectively improves the matching accuracy of the robotic arm to the GUI control icons on the mobile device screen.**

***Keywords*—visualization testing, feature extraction, image matching, Siamese, RepVGG network***

## I. INTRODUCTION

With the continuous development of intelligent manufacturing, computer vision technology plays a key role in modern industry, making it easy to obtain digital image information in smart factories [1][2][3]. These technologies are used in a wide range of applications, including surveillance cameras in public areas of factories [4], visual recognition systems on industrial production lines [5], and in the manufacturing process of precision instruments [6]. Promoting these advanced technologies has dramatically improved production efficiency and product quality, and image-matching technology is the basis for realizing these applications. In intelligent manufacturing, comparing the similarity of two images has been the focus of research and a key component of basic research. This technique has been widely used in machine vision and image processing, such as workpiece machining, product quality inspection, part localization, and image analysis tasks [7][8]. The image matching technique has important research significance and practical application value on the task of mobile device GUI screen testing on factory production lines.

In the intelligent manufacturing production line, the traditional template matching algorithm in matching mobile screen GUI controls is prone to phenomena such as complex backgrounds, different screen resolutions, mobile device tilting, and weak processing capability of icon scaling transformations leading to mismatching during testing [9]. Feature point-based matching algorithms may face noise interfering with the extraction of features, sensitivity to lighting changes, local dependency, and other factors leading to matching failure [10]. The limitations of these traditional matching methods, which are inefficient in handling large-scale script testing tasks, drive people to seek better methods to enhance matching accuracy and efficiency.

With the wide application of deep learning technology, image-matching algorithms based on deep learning have gradually replaced traditional methods, especially in the field of intelligent manufacturing production lines [11][12][13]. These deep learning models can extract deeper features in the image, significantly improving the matching accuracy [14]. Deep learning has been shown to outperform traditional techniques in areas such as industrial inspection, quality control, and automated assembly [15][16]. Therefore, applying deep learning image matching algorithms to the testing of GUI controls for mobile devices can improve the accuracy and efficiency of product quality in automated testing, which provides an important practical value for the development of intelligent manufacturing [17][18].

In summary, during the visualization GUI testing process, the actual position of the GUI controls of mobile devices is located by industrial cameras and robotic arms. In order to solve the poor matching effect and mismatching phenomenon caused by the mobile device screen being susceptible to the interference of complex environment as well as the simple structure and inconspicuous features of some GUI controls, a matching framework for the visualization GUI test is proposed. The framework first designs a targeted feature extraction module based on GUI controls' shape and structure characteristics. Subsequently, an improved Siamese network model is utilized to match the processed icons with template icons, thus realizing the accurate positioning of the robotic arm to the GUI controls on the mobile device's screen.

The main contributions of this paper are as follows:

(1) By cutting and storing the GUI controls in the public dataset Rico and expanding them. Then, a GUI_Icon control icon dataset is constructed using image enhancement techniques such as rotation, blurring, and exposure.

(2) Aiming at the limitations of mobile device GUI controls in feature extraction, a module for effectively extracting feature information of control icons on the screen is proposed.

(3) The Siamese network model is improved by replacing the RepVGG network backbone to enhance the model's inference speed. Meanwhile, the CA attention module is applied to strengthen the extraction effect of control icon features, and the GELU activation function is introduced to improve the learning speed and accuracy of the network.

The rest of the paper is structured as follows: Section II discusses research related to matching algorithms based on traditional image matching and deep learning. Section III describes the matching process for control icons in visual automated testing and the research methodology used in the article. In Section IV, comparative experiments and analysis are performed. In Section V, a summary of the article is presented.

## II. Related work

In recent years, image-matching algorithms have played an important role in computer vision and image analysis, especially in the application scenario of detecting GUI control icons of mobile devices in industrial smart manufacturing. Such algorithms are not only the key technology to realize industrial robots to perform workpiece sorting work and object classification but also play an important role in anomaly detection and target recognition tasks. Therefore, the research on image-matching tasks has significant academic and practical significance. When an industrial camera is used to discriminate whether an image on the screen of a mobile device is similar to a target image in a factory environment, external objective factors such as object occlusion, illumination variations, and different shooting angles can significantly affect the final appearance of the image. To cope with these challenges, many descriptors on image features and Siamese network matching algorithms have been developed, and their proposals have greatly advanced the development of computer vision. However, these traditional manual descriptors often struggle to achieve optimal results when dealing with the abovementioned influencing factors. Therefore, how to effectively solve the interference of these external factors has become the core topic of image-matching technology research.

In the context of industrial intelligent manufacturing, we can utilize industrial cameras to directly acquire image datasets under the scene. Through this, we can automatically learn the features of image blocks to construct a similarity function. Based on this similarity function, we can more accurately discriminate the similarity of two images and then realize more efficient image matching when detecting GUI control icons of mobile devices. This process improves the accuracy of image matching and provides new ideas for image processing applications in intelligent manufacturing.

### A. *Feature point based image matching algorithm*

The method of detecting images by locating feature key points and then pre-processing the target has been widely used in intelligent manufacturing assembly lines. Harris corner detection algorithm proposed by Harris et al. obtains corner features by sliding the window on the image. However, it is ineffective in extracting feature points with scale variation, and the threshold value affects the number of features [19]. Lowe proposed that the SIFT algorithm has good scale and rotation invariance [20]. The SURF algorithm proposed by Mikolaj-czyk et al. is much faster than the SIFT algorithm in feature extraction [21]. Other mainstream methods for keypoint detection are FAST [22], BRIEF [23], ORB [24], KAZE [25], etc.

Aiming at the various distortions that may occur when industrial cameras acquire the screen, such as colour distortion, moiré, screen inversion, and sensor noise, Gan et al. proposed an entropy-weighted Harris corner detection algorithm based on entropy-weighted Harris corner detection algorithms in order to extract feature points in the screen that are rich in texture features and have high robustness. Subsequently, the SIFT algorithm is used to assign orientations to these feature points to construct and filter out non-overlapping feature regions [26]. Fang et al. In order to prevent the loss of details in the screen image due to special aberrations induced by the lens during the shooting process, an intensity-based scale-invariant feature variation algorithm is used to enhance the intensity of the detection of the key points to locate the target region [27] accurately. Li et al. feature regions into the SuperPoint keypoint detector to get the key points with stable feature icons on the screen [28]. Bai et al. use the ORB feature point algorithm to quickly find the feature points of the target image on the screen at different shooting distances and angles [29]. Although these key point localization methods allow industrial cameras to reduce the inaccuracy of localization caused by external interference on the screen when locating icons on the screen, GUI control icons need further improvement of the icon localization methods through the characteristics of the controls because of their diverse structure and scattered distribution in the screen.

### B. *Neural network based matching algorithm*

As convolutional neural networks gradually occupy an important position in computer vision, more and more neural networks are used to realize the extraction and matching of icon images. Han et al. proposed the Matchnet algorithm by inputting images of the same size, and the distance obtained is used to calculate the similarity of two of their feature vectors in a learning manner [30]. Zagoruyko et al. utilized a combination of twin, dual-channel, and dual-stream networks to propose a DeepCompare network to improve the matching of two image blocks [31]. In the DeepDesc method proposed by Simo-Serra et al., a hard sample mining strategy is trained using a twin network structure to obtain a 28-dimensional feature vector. Although DeepDesc performs superiorly in image matching, it incurs significant computational costs [32]. Tian et al. trained L2-net with the number of L2 norms of the extracted features by employing a progressive sampling strategy, which performed better than DeepDesc.Balntas et al. trained the network using a patch triad containing positive and negative pairs. They introduced a PN-Net network that outperforms MatchNet regarding performance and memory advantages over MatchNet and DeepCompare [33].

These methods use twin neural networks to compare and match two images. Compared with traditional image-matching algorithms, this convolutional neural network-based algorithm excels in one-to-one matching performance. However, the dataset acquisition and training processes require much work, and the results of matching target images with template images in complex environments are often unsatisfactory. In the study of this paper, the direct application of neural networks to match GUI controls with template images on a mobile device's screen may lead to many false matches, which is unacceptable in the smart manufacturing industry. Based on the complementary nature of keypoint localization and neural network feature matching, we use key points to accurately locate all the GUI control icons on mobile devices in complex environments and perform image matching with the help of well-performing neural networks, which is the research direction of this paper.

## III. Proposed Method

Aiming at the problems of poor recognition accuracy due to the influence of factory environment interference when the industrial robotic arm performs the visualization test task, as well as the poor effect of the matching algorithm in the traditional test task. This paper proposes a testing framework, as shown in Figure 1. It mainly includes three modules: an image preprocessing module, an image matching module, and an action execution module. When the industrial robotic arm receives the script command of the test task, it will first obtain the screen image of the mobile device on the desktop through the industrial camera and split the screen image into a GUI control set through the image preprocessing module. The robotic arm will match the extracted control icon set with the template control icon set stored in the script command in the image matching module and then match the control icon corresponding to the camera input image in the action execution module; finally, in the action execution module, it will match to the control icon corresponding to the camera input image in the action execution module. Finally, in the action execution module, the control pixel coordinates obtained from the image matching module are converted to actual coordinates, and then combined with the corresponding action commands in the script commands, the industrial robotic arm is allowed to carry out the expected test tasks.

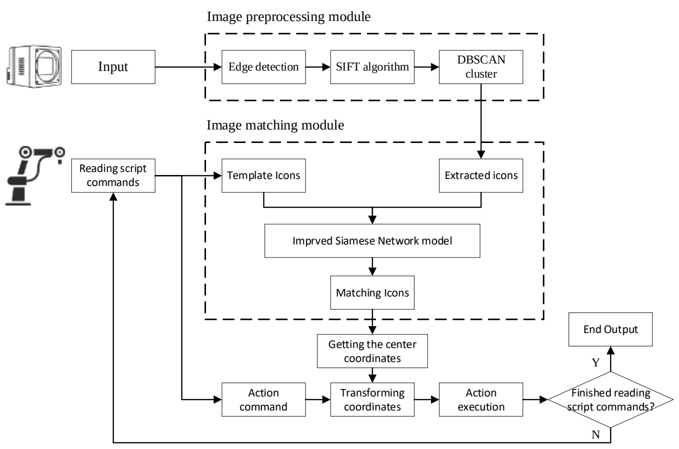

Figure 1 Visual automation test flow.

The main innovations of this paper will be further described next in terms of the image preprocessing module and the image matching module. As shown in Figure 2.

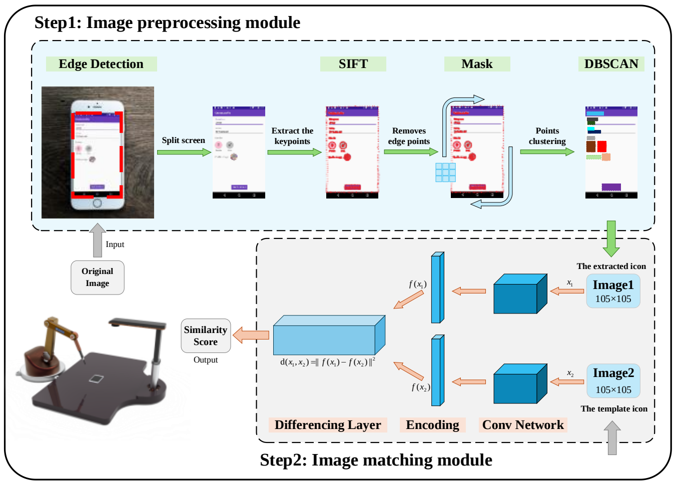

Figure 2 GUI control icon matching framework.

Wherein the image preprocessing module takes the screen image acquired by the camera and first extracts the feature information within the screen of the mobile device individually through an edge detection algorithm, then extracts all the feature points in the picture using the sift algorithm, then eliminates the redundant feature points at the edges of the screen using mask processing, and finally divides all the feature points into several different regions using the DBSCAN clustering algorithm, and divides Finally, the DBSCAN clustering algorithm is used to divide all the feature points into several different regions. Each region is divided into a separate image, the corresponding GUI control icon on the screen.

The image matching module uses the template control icons and the set of on-screen control icons collected in the image preprocessing module as inputs, uses the improved Siamese network model to perform image matching, and finally obtains the positions of the images larger than the threshold range, which are the actual pixel coordinate positions matched on the mobile device.

## A. *Image Extraction Methods*

### *1) Improved edge detection algorithm*

When an industrial camera shoots a mobile device's screen, it will also shoot the extra background outside the screen. The external background is easy to follow up on the screen GUI controls to do feature extraction caused by interference, so it is necessary to do image processing before the screen image edge processing. The traditional Canny algorithm is prone to distortion and loss of edge information when processing mobile device screens. Dual threshold detection is ineffective in connecting edges, and it is difficult to cut out the complete shape of the screen. Due to the difference between the bright light of the mobile device screen and the external background, the image can be divided into foreground and background parts by the grey value. This paper proposes a Canny detection algorithm based on Otsu threshold segmentation to enable industrial cameras to cut out the screen information completely during the shooting process for subsequent image processing.

The specific flow of the threshold segmentation algorithm is shown in Figure 3. Firstly, the captured image is divided into two parts $B_1$、$B_2$ by the gray value using the defined segmentation threshold$T_0$. By calculating their average gray scale $g_1$、$g_2$ and total pixel point ratio $e_1$、$e_2$, the maximum value of inter-class variance σ of all pixels is obtained as the segmentation threshold $T_0$. the formula is as follows:

$$\sigma = e_1 * e_2(g_1 - g_2)^2 \quad (1)$$

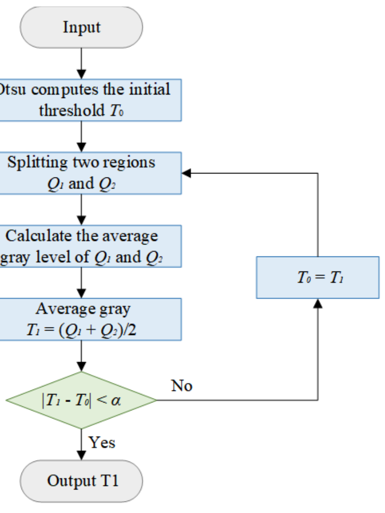
Input
Otsu computes the initial threshold $T_0$
Splitting two regions $Q_1$ and $Q_2$
Calculate the average gray level of $Q_1$ and $Q_2$
Average gray $T_1 = (Q_1 + Q_2)/2$
$|T_1 - T_0| < \alpha$
No
$T_0 = T_1$
Yes
Output T1

Figure 3 Flow of Ostu threshold segmentation algorithm.

Second, the image is divided into two regions $Q_1$ and $Q_2$ using the threshold value $T_0$, and the average gray level of these two regions is calculated, where the number of pixel points corresponding to each gray level i is $n_i$ and the gray level is set to K. And the average value of $Q_1$ and $Q_2$ is treated as the new threshold value $T_1$.

$$Q_1 = \sum_{i=0}^{T_0-1} in_i / \sum_{i=0}^{T_0-1} n_i \quad (2)$$

$$Q_2 = \sum_{i=T}^{K} in_i / \sum_{i=T}^{K} n_i \quad (3)$$

$$T_1 = \frac{Q_1+Q_2}{2} \quad (4)$$

If the difference between the absolute values of $T_1$ and $T_0$ obtained is less than within the error range, then $T_1$ is the optimal segmentation threshold; otherwise, make $T_0 = T_1$ and continue to perform the above steps until the conditions are met.

Finally, the optimal threshold $T_1$ is used for image segmentation, and the high and low thresholds $T_{max} = T_1$ and $T_{min} = 0.4T_1$ are set. the image edge pixels are discriminated, and if $T_{max}$ is greater than then it is recorded as a strong edge, and the pixel is set to 255; if it is less than $T_{min}$ then it is recorded as a weak edge, and the pixel is set to 0; and if it is between the two then it is discriminated accurately again. After iterating through the entire region, if there are one or more strong edge points in the 8-connected region, they are also labeled as strong edge points. Multiple iterations of this process are performed until no new pixels are classified as strong edges and then the iteration is stopped.

In this paper, the Canny algorithm improved by Otsu makes the segmented binarized image richer under the bright light difference between the screen and the outside background. The ability to accurately segment the required mobile device screen in the whole picture obtained from shooting reduces the burden of extracting the GUI control image using feature points afterwards. It lets the attention focus on the screen instead of the non-working area, increasing the matching accuracy.

### *2) Sift feature point extraction*

The SIFT algorithm has good noise, brightness, rotation, and translation invariance. It can extract the feature points on each GUI control icon well, even if the screen is shifted during the screen control feature extraction task. The extraction of feature points based on the SIFT algorithm is mainly realized by constructing a Gaussian difference pyramid, and its main steps are as follows:

First, a Gaussian pyramid is constructed to obtain images at different scales. Next, two neighbouring Gaussian space images at the same scale are subjected to a subtraction operation to generate a Gaussian difference pyramid image. Notably, the upper image of the Gaussian difference pyramid is obtained by downsampling the penultimate third image of the lower image. The Gaussian difference pyramid is calculated as:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/(2\sigma^2)} \quad (5)$$

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad (6)$$

Where σ is the degree of smoothing of the image, reflecting the degree of Gaussian blurring, (x, y) represents the spatial coordinates, G is the Gaussian kernel function, L is the scale space of the image, and I is the original input image.

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \quad (7)$$

In this formula, k denotes the scale factor of two neighbouring Gaussian scale spaces. This function enables feature points in the scale space to be effectively detected. We extract feature points with good scale transformation and robustness by constructing a multi-scale space pyramid of the image. For each detected feature point, we judge whether the point is an extremely small or extremely large value in the 3×3×3 neighbourhoods of the scale layer where it is located and the two scale layers above and below it. It is regarded as a feature point if the point is extremely small or extremely large.

Finally, the extreme points are filtered. Extreme points that do not describe the GUI control are filtered out by a curve that fits the DoG function to eliminate non-compliant points. The retained feature points are mainly concentrated on the control icons in preparation for the subsequent clustering.

### 3) DBSCAN clustering algorithm

Today's GUI control icons are rich in variety, not only the standard rectangle, circle, and other graphics. Meanwhile, when UI designers design the controls on the screen, the distribution of its controls is scattered and spaced to make the interface clearer and more logical. To aggregate the feature point information on the GUI controls obtained by the SIFT algorithm into small pieces of icons and input them into the next image-matching algorithm, this paper adopts the DBSCAN clustering algorithm for the processing of the feature point information.DBSCAN is an algorithm for spatial clustering based on the density information of data points, proposed by Ester et al. in 1996 [34], and it can classify the data with high density into clusters and form arbitrary shapes. Into clusters, forming clusters of arbitrary shapes. The basic idea is to cluster based on the density relationship between the samples and identify the core, boundary, and noise samples by adjusting the neighbourhood range around the samples and the minimum number of samples. The steps of the algorithm are as follows:

Algorithm 1 DBSCAN algorithm steps

```
1:  Input Dataset D, neighborhood radius Eps, minimum sample count
    MinPts
2:  Output Clustering result
3:  Initialize Mark all sample points as unvisited
4:  Initialize C = empty list of clusters
5:  for each sample point A ∈ D then
6:    if A is unvisited then
7:      Mark A as visited
8:      N = calculateNeighborhood(A, Eps)
9:      if |N| < MinPts then
10:       Mark A as a noise point
11:     else
12:       Create a new cluster C_i
13:       Add A to C_i
14:       for each point B ∈ N then
15:         if B is unvisited then
16:           Mark B as visited
17:           N' = calculateNeighborhood(B, Eps)
18:           if |N'| ≥ MinPts then
19:             Add N' to N
20:         if B is not in any cluster then
21:           Add B to C_i
22:       Add C_i to C
23: Return C
```

Step 1: Starting from the unvisited sample point A, check all the sample points within its neighbourhood and determine whether they meet the condition of the core point, i.e., the number of samples in the neighbourhood M must be greater than or equal to the minimum number of samples MinPts.

Step 2: If sample point B meets the condition of core point, it is selected as the core point of the new cluster, and all sample points within its neighbourhood are grouped into this cluster. Meanwhile, recursively expand the sample points in the neighbourhood as new core points until no new core point can be found.

Step 3: If sample point A does not satisfy the condition of the core point but lies within the neighbourhood of other clusters, it is regarded as a boundary point and added to the corresponding cluster.

Step 4: If the sample point neither satisfies the condition of the core point nor lies within the neighbourhood of any other cluster, it is marked as a noise point.

Step 5: Continue the above steps until all the sample points have been visited.

The advantages of DBSCAN include the ability to recognize clusters of arbitrary shapes, high robustness to noise points, no need to specify the number of clusters in advance, and the ability to effectively deal with control icon information with a large change in density. It can play a good clustering effect when dealing with GUI controls with obvious features and organized distribution on the screen.

## B. Improved Siamese network

The structure of the Siamese network is shown in Figure 4. It consists of two identical and shared-parameter convolutional neural networks, including convolutional layers, activation functions, pooling layers, and corresponding loss functions. When performing the GUI control matching task, these two convolutional neural networks will extract features from the input interface control images. The extracted features are fed into a decision module consisting of one or more fully connected layers, which are compared with the labels to generate a similarity score of the matched controls. The loss function used in the learning process of this Siamese network can be a ternary loss, binary cross-entropy loss, or contrast loss, among others. After the output of the fully connected layer, the decision layer converts the image features into one-dimensional feature vectors, and the loss function maps the one-dimensional vectors of the corresponding branches into Euclidean space. By narrowing the Euclidean distance between correct matches and enlarging the distance between incorrect matches, this strategy can effectively determine the similarity between the input GUI controls.

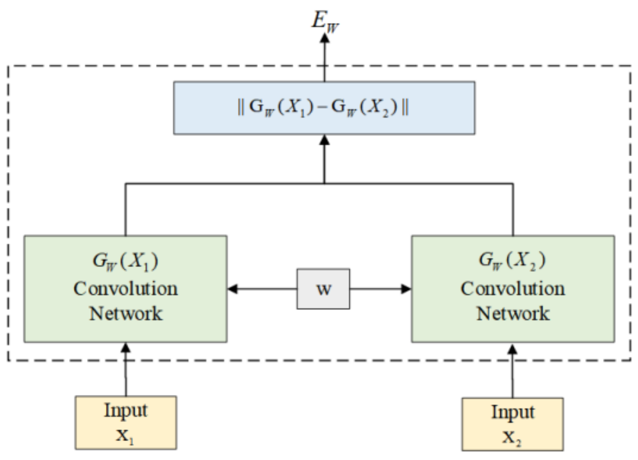
$E_W$
$\| G_W(X_1) - G_W(X_2) \|$
$G_W(X_1)$ Convolution Network
w
$G_W(X_2)$ Convolution Network
Input $X_1$
Input $X_2$

Figure 4 Siamese network structure.

This paper proposes a Siamese-based fusion feature icon-matching algorithm for matching the acquired GUI control image blocks. The RepVGG network backbone is replaced to improve the model's inference speed; the CA attention module is introduced to strengthen the extraction effect of control icon features; and the GELU activation function is added to improve the network's learning speed and accuracy. The overall network can accurately match the target icon in the factory interference environment when GUI controls are matched on the mobile device screen.

### *1) RepVGG backbone network*

RepVGG network is a multibranch network structure based on VGG networks proposed by Ding et al. in 2021 [35]. The core of the RepVGG network lies in effectively decoupling the training and inference networks through structural reparameterization (i.e., merging the trained multibranch structure into a single-branch structure during inference). This feature not only makes full use of the advantages of multibranch networks in feature extraction but also achieves high speed and low memory consumption when inference is deployed, which provides a good performance basis for applications on mobile devices.

In the training phase, the RepVGG network mainly consists of 3×3 convolutional kernels, 1×1 convolutional kernels, and directly connected branches. Adding directly connected branches and 1×1 convolutional branches in parallel allows information at different image scales to be extracted and effectively fused, thus enhancing the model's representational capability. The training structure is schematically shown in Figure 5.

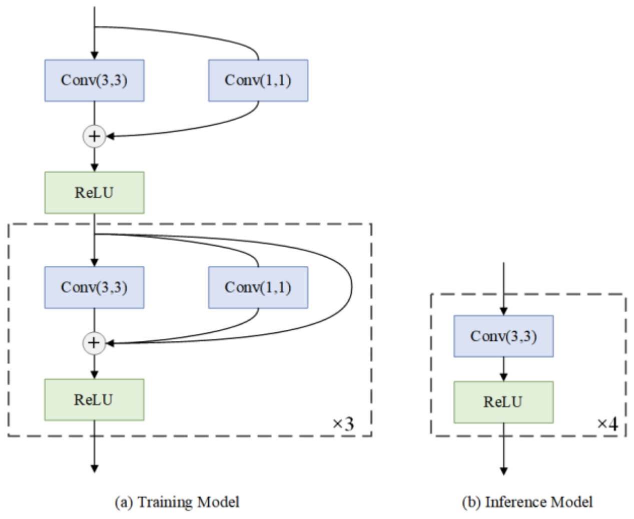

Figure 5 RepVGG network structure.

In the inference phase, the RepVGG network fuses the 1×1 convolution and direct-connected branches from training into a 3×3 convolution through structural reparameterization, resulting in a branchless unilinear structure consisting mainly of a 3×3 convolution and an activation function. This transformation allows the network not only to retain the parameter weights obtained in multibranch training but also to utilize the single linear structure to accelerate the inference speed of the model in the deployment inference phase. Meanwhile, the deep optimization of the 3×3 convolution based on the NVIDIA deep neural network library further improves the detection speed of the RepVGG network in the inference phase. The structural reparameterization in the inference stage mainly includes the fusion of the convolutional kernel with the Batch Normalization (BN) layer, the integration of the 1×1 convolution into the 3×3 convolution, and the integration of the direct-connected branches into the 3×3 convolution. With these optimizations, the RepVGG network demonstrates good performance and efficiency for applications on mobile devices.

### *2) CA Attention Module*

In factory environments, inspection objects are often disturbed by outside light variations, equipment jitter, and other factors. These effects can lead to insufficient features of the screen controls acquired by the camera, thus making it difficult for the elements to be effectively recognized by the machine. Existing attention modules focus on global features and ignore scattered non-important feature information. This may lead to the loss of salient features for elements originally affected by the environment, thus decreasing the recognition accuracy instead of increasing it, which defeats the original purpose of introducing the attention module. Therefore, in order to better extract features and enhance the weak feature information affected by interference, this paper introduces the coordinate attention (CA) mechanism [36], which can focus on the inter-channel information of the feature map and the spatial location information at the same time, and its module structure is shown in Figure 6.

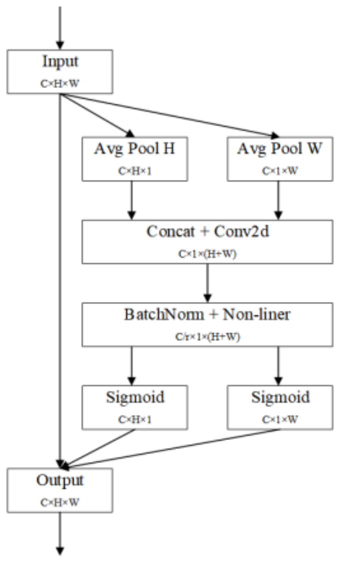

Figure 6 Structure of CA attention module.

First, each channel is encoded along the horizontal and vertical coordinates using the average pooling kernel of size $(H, 1)$ and $(1, W)$. The average pooling outputs $e_c^h(h)$、$e_c^w(w)$ are obtained for the two sets of cth channel representing height h and width w, respectively, with the following correlation equations:

$$e_c^h(h) = \frac{1}{W}\sum_{0\leq i\leq W} x_c(h, i) \tag{8}$$

$$e_c^w(w) = \frac{1}{H}\sum_{0\leq j\leq H} x_c(j, w) \tag{9}$$

Then, through convolution and splicing operations, the obtained feature information is obtained after batch normalization and nonlinear activation function operations to obtain the feature f. It is then divided into tensors $f_h$ and $f_w$, and after convolution $\mathrm{Conv}(\cdot)$ and activation function $\mathrm{Sigmoid}(\cdot)$, it generates weights $w_h$ and $w_w$ along the two directions of height and width, with the following formulas:

$$w_h = \mathrm{Sigmoid}(\mathrm{Conv}(f_h)) \tag{10}$$

$$w_w = \mathrm{Sigmoid}(\mathrm{Conv}(f_w)) \tag{11}$$

Finally, the original feature information $w_x$ is multiplied element-by-element with weights $w_h$ and $w_w$ to obtain the output feature $w_y$.

$$w_y = w_x * w_h * w_w \tag{12}$$

*3) GELU activation function*

In RepVGG networks, the Rectified Linear Units (ReLU) activation function effectively mitigates the problem of vanishing or exploding gradients as the neural network deepens. However, the limitations of the ReLU activation function are also not negligible, especially when dealing with GUI control features. When the input is less than zero, ReLU will set the output to zero directly, which leads to the corresponding neurons being permanently disabled, thus affecting the convergence and feature extraction ability of the network model. Therefore, to address this problem, Gaussian Error Linear Units (GELU) are chosen as the activation function in this paper [37].

The GELU activation function is applied to the CA module as a nonlinear unit. Its advantage is that GELU is differentiable at the origin and introduces the idea of stochastic regularization in its definition. This feature enables the activation operation to establish a stochastic connection between the input and the output, effectively avoiding the situation where the neurons are set to zero, especially in scenarios where complex features need to be extracted, thus improving the learning speed and accuracy of the network. Using the GELU activation function, the network can better utilize its structure and optimize its performance, providing a solid foundation for feature extraction and model convergence.

## IV. EXPERIMENTAL RESULTS AND ANALYSIS

### A. *Experimental setup and data set introduction*

The experimental platform used is a Windows operating system. The CPU is an Intel Core i7-12700, and the GPU is an Nvidia GeForce RTX3060 with 6G video memory and 16G RAM. Deep learning frameworks Pytorch 1.12.1, Python 3.8, and CUDA 11.2 were used for training in the experimental environment. To prevent model overfitting, the experiments were conducted on a setup where the network was trained for 200 rounds using the Adam optimizer with an initial learning rate of 1e-2 for the backbone network and a batch size 32.

Due to the lack of readily available GUI control icon datasets in the mobile device visualization test task, this paper adopts the method of manually photographing and processing existing publicly available datasets to construct a dataset that meets the experimental requirements. Specifically, manual shooting is used to shoot the mobile device screen using an industrial camera in the current experimental environment, and the acquired screen image is subsequently cut to extract the GUI control icons from the screen. On the other hand, the existing publicly available dataset Rico provides screenshots for mobile device screens, so it is necessary to filter out the portion that meets the experimental requirements and cut and save the GUI controls in each screenshot.

Combining the GUI control icons obtained from these two methods, we performed data enhancement and preprocessing of the images. The data enhancement techniques include image tilting, light intensity adjustment, noise addition, and contrast adjustment. The dataset produced in this process covers common types of GUI controls in mobile devices, such as switches, buttons, text, images, and input boxes, totalling 500 different images. Each GUI control icon was adjusted to 22 images after data enhancement, and finally, all the images were resized uniformly to 105×105 pixels.

To ensure the robustness of the experiments, this paper also uses the Omniglot dataset, which is similar to the GUI control, for training and validation. This dataset contains different alphabets of various languages with 1623 categories, 20 training samples for each category, and the size of each image is also 105×105 pixels. Eventually, the two datasets are divided into training and validation sets in the ratio of 9:1 to support the subsequent experimental analysis. Figure 7 shows some of the GUI_Icon dataset images constructed in this paper.

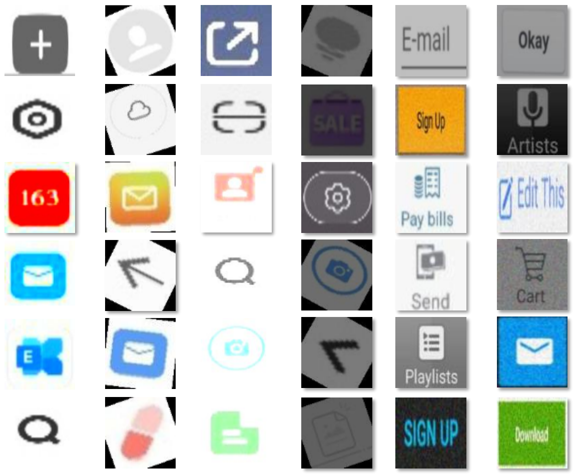

Figure 7 GUI_Icon dataset.

### B. *Evaluation indicators*

The evaluation metrics of the model are mainly used as Average Accuracy (Accuracy, Acc), AUC (Area Under the ROC Curve), and ROC Curve (Receiver Operating Characteristic Curve). Among them, the average accuracy is the proportion of icon samples correctly classified in the provided GUI dataset test set, which reflects the overall classification effect of the model. The ROC curve is used to evaluate the model's overall performance in recognizing GUI controls, which demonstrates the recognition effect under different thresholds and is, therefore, also known as the Subject Operating Characteristic Curve. By observing the shape of the ROC curve, the performance of the model can be visually assessed. On the ROC curve, the horizontal axis represents the False Positive Rate (FPR), while the vertical axis represents the True Positive Rate (TPR). When plotting the ROC curve, it is possible to show the model's performance under each threshold by setting different thresholds, which helps to gain a deeper understanding of the model's recognition ability. The formulas for TPR, FPR, and Average Accuracy Rate are as follows:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP+FN}} \tag{13}$$

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP+TN}} \tag{14}$$

$$\mathrm{Acc} = \frac{\mathrm{TP+TN}}{\mathrm{TP+TN+FP+FN}} \times 100\% \tag{15}$$

Wherein denotes the number of samples that were correctly identified as positive examples, denotes the number of samples that were incorrectly identified as negative examples, denotes the number of samples that were incorrectly identified as positive examples, and denotes the number of samples that were incorrectly identified as negative examples.

## C. Comparison with other models

To further verify the effectiveness of the improved structure, under the same experimental environment, we selected some classical convolutional neural networks, ResNet, Densenet, VGG, and RepVGG, by training them on the GUI_Icon dataset and Omniglot dataset. The final results are shown in Table 1. Our improved structure has high accuracy under both datasets, reaching 98.6% and 95.5%, respectively, at least 1.9% and 1.7% better than other models. Meanwhile, the parameters and computation of the model are slightly increased compared to RepVGG but much reduced compared to other models.

Table 1 Comparison of evaluation indicators of different network models.

| **Models** | **Param.(M)** | **GFLOPs** | **ACC** | |
|---|---|---|---|---|
| | | | ***GUI_Icon*** | ***Omniglot*** |
| ResNet18 | 11.6 | 18.2 | 95.4 | 92.8 |
| ResNet34 | 21.7 | 36.7 | 96.1 | 93.1 |
| Densenet | 20.1 | 43.9 | 94.8 | 92.1 |
| VGG13 | 133.0 | 113.0 | 94.2 | 91.7 |
| VGG16 | 138.3 | 154.7 | 94.8 | 92.2 |
| RepVGG | 8.3 | 16.7 | 96.7 | 93.8 |
| Ours | 10.3 | 17.1 | 98.6 | 95.5 |

## D. Ablation experiments

To better demonstrate the performance of our improved model in practice, it is evaluated on the GUI dataset through ablation experiments, shown in Table 2.

First, a single module is introduced for comparison. By introducing the RepVGG network, not only the detection accuracy is increased by 1.9% compared to the original model, but also 0.7ms reduce the detection speed, and there is a significant reduction in the parameters and computation amount of the model; the CA attention mechanism is added to the original model, which improves the detection accuracy and also increases the inference time of the model; after replacing the activation function in the model. After replacing the activation function in the model, the detection accuracy is increased to 96.9%, and the inference time is not much different from the original model. Secondly, after introducing the RepVGG network and CA module at the same time, the detection accuracy of the model increased to 98.2%, and the inference time reached 6.8ms with a slight increase in parameters and computation; the detection accuracy and inference time of the model also increased after introducing the RepVGG network and the GELU activation function. Introducing the three modules into the final network simultaneously, the detection accuracy reaches 98.6%, and 0.6ms reduces the inference time compared to the original model for a single image. The model's parameters and computation are reduced by 92.5% and 88.9%, respectively. The experimental results show that the improved model proposed in this paper has excellent image-matching accuracy, but its detection speed, parameters of the model, and computation are also improved.

Table 2 Comparison of ablation experiments based on elevated accuracy module.

| **Num** | **Model** | | | **Param (M)** | **GFLOPs** | **ACC** | **Time (ms)** |
|---|---|---|---|---|---|---|---|
| | ***RepVGG*** | ***CA*** | ***GELU*** | | | | |
| 0 | | | | 138.3 | 154.7 | 94.8 | 7.4 |
| 1 | √ | | | 8.3 | 16.7 | 96.7 | 6.1 |
| 2 | | √ | | 138.9 | 155.1 | 97.8 | 7.7 |
| 3 | | | √ | 138.4 | 154.8 | 96.9 | 7.5 |
| 4 | √ | √ | | 10.2 | 17.0 | 98.2 | 6.8 |
| 5 | √ | | √ | 8.4 | 16.8 | 97.7 | 6.1 |
| 6 | √ | √ | √ | 10.3 | 17.1 | 98.6 | 6.8 |

## E. Comparison of multiple matching methods

To verify the matching effect of the matching method proposed in this paper in practice, we use a dataset containing 500 screen images to experimentally compare and verify with the corresponding template images by comparing with the template matching algorithm as well as common feature point matching algorithms, and the results are shown in Table 3.

The template matching algorithm outperforms the classical feature point matching algorithms SIFT and ORB regarding the correctness and total time consumed for matching. Super Point [38], Super Glue [39], and LoFTR [40] matching algori-thms are commonly used for 3D image matching, and they are applied to the dataset of this paper to perform the matching with good matching results. However, they are much slower than the other matching algorithms. However, it is much slower than other matching algorithms. At the same time, the GUI controls in this dataset are not rich in feature points, and the algorithm based on feature point matching will have the phenomenon of missing matching in the matching process, which leads to the matching box being selected to the corresponding icon but cannot cover the whole icon normally. The matching box of template matching is matched according to a fixed size, so the number of missed matches is less, and the data in the table can clearly show this phenomenon. The matching method proposed in this paper is much more accurate than other existing methods, reaching 94.4%, while the total time consumed for matching is relatively short. Our proposed matching method is well-suited for visual screen testing in terms of high accuracy and speed.

Table 3 Comparative experiments of different matching algorithms.

| **Methodology** | **Match** | **Mismatch** | **Miss Match** | **Cost Time/s** |
|---|---|---|---|---|
| Template | 441 | 42 | 17 | 2.23 |
| SIFT | 423 | 32 | 45 | 3.85 |
| ORB | 404 | 36 | 60 | 1.97 |
| Super Point | 457 | 15 | 28 | 4.93 |
| Super Glue | 433 | 24 | 43 | 4.53 |
| LoFTR | 449 | 19 | 32 | 4.36 |
| Ours | 472 | 11 | 17 | 2.56 |

### F. *Experimental validation and analysis*

To verify the performance of the matching method in this paper in the test task, we use an industrial robotic arm and industrial camera to simulate a more realistic visualization test environment and use the connected computer platform for task processing. As shown in Figure 8, the industrial camera photographs the mobile device on the desktop in a dark environment, and the acquired images are subjected to edge detection, key point extraction, edge masking, and clustering algorithms in turn; the segmented set of control icons is matched with the template icons provided by us, and the results of the experiments are shown in Figure 9, which clearly shows that the matching method proposed by us can accurately identify the target icons. Combined with the experimental data in the previous section, our inference speed has also been improved to meet the real-time detection performance in the intelligent manufacturing production line.

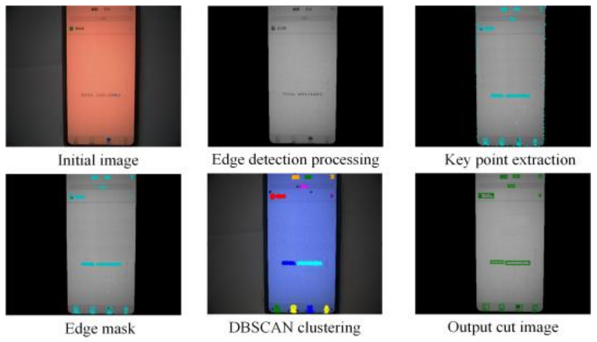

Figure 8 Effect of screen image feature extraction.

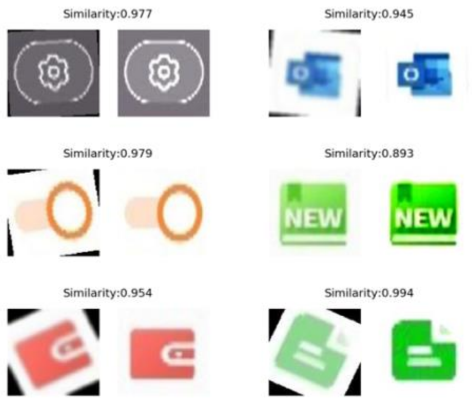

Figure 9 Matching similarity of control icons.

## V. CONCLUSION

To better perform vision automation test tasks on intelligent manufacturing production lines, the problem of frequent mismatches leading to test failures in executing scripted test tasks by industrial robots has been reduced. At the same time, it is also considered that the factory environment may interfere with the acquisition of screen images from mobile devices. To this end, this paper proposes an image-matching method based on a twin network, where all the GUI control icons in the input image are effectively extracted by preprocessing the images and combined with the template icons in the script commands as image inputs to the twin network. Among them, the improved twin network improves the detection speed while increasing the matching accuracy by introducing the RepVGG network. The added CA attention module can extract more effective feature information from both spatial and channel directions simultaneously, improving detection accuracy. Adding the GELU activation function makes the model faster during training and performs well in matching. After ablation experiments and comparative analysis with other models, our improved model increases the speed of inference while improving the detection accuracy of the model. It is well suited to be deployed to perform automated testing tasks on manufacturing production lines.

## ACKNOWLEDGMENT

This work was supported by the Communication Soft Science Program of Ministry of Industry and Information Technology, China (No.2022-R-43), the Natural Science Basic Research Program of Shaanxi (No.2021JQ-719), the Special Scientifc Research Program of Education Department of Shaanxi(No.22JK0562), the Youth Innovation Team of Shaanxi Universities “Industial Big Data Analysis and Intelligent Processing,” and the Special Funds for Construction of Key Disciplines in Universities in Shaanxi.

## REFERENCES

[1] B. Wang, F. Tao, X. Fang, C. Liu, Y. Liu, T. Freiheit, Smart Manufacturing and Intelligent Manufacturing: A Comparative Review, Engineering 7 (6) (2021) 738-757, https://doi.org/10.1016/j.eng. 2020.07.017.

[2] G. Nain, K. Pattanaik, G. Sharma, Towards edge computing in intelligent manufacturing: Past, present and future, Journal of Manufacturing Systems 62 (2022) 588-611, https://doi.org/10.1016/j.jmsy. 2022.01.010.

[3] S. Sahoo, C. Lo, Smart manufacturing powered by recent technological advancements: A review, Journal of Manufacturing Systems 64 (2022) 236-250, https://doi.org/10.1016/j.jmsy.2022.06.008.

[4] J. Dai, Q. Li, H. Wang, L. Liu, Understanding images of surveillance devices in the wild, Knowledge-Based Systems 284 (2024) 111226, https://doi.org/10.1016/j.knosys.2023.111226.

[5] L. Xia, J. Lu, Y. Lu, H. Zhang, Y. Fan, Z. Zhang, Augmented reality and indoor positioning based mobile production monitoring system to support workers with human-in-the-loop, Robotics and Computer-Integrated Manufacturing 86 (2024) 102664, https://doi.org/10.1016/j.rcim. 2023.102664.

[6] A. Adeleke,INTELLIGENT MONITORING SYSTEM FOR REAL-TIME OPTIMIZATION OF ULTRA-PRECISION MANUFACTURING PROCESSES, Engineering Science & Technology Journal 5(3) (2024) 803-810, https://doi.org/10.51594/estj.v5i3.904.

[7] M. González, A. Rodríguez, U. López-Saratxaga, O. Pereira, L. Norberto, Mikel González and Adrián Rodríguez and Unai López-Saratxaga and Octavio Pereira and Luis Norberto, Journal of Manufacturing Systems, 74 (2024) 41-54, https://doi.org/10.1016/j.jmsy.2024.02.014.

[8] Q. Fu, H. Yan, Z. He, W. Li, W. Zang, C. Zhu, Cheng, Research and Design of Machine Vision-based Workpiece Defect Sorting Robot, 2024 International Conference on Intelligent Computing and Robotics (ICICR) (2024) 212-219, 10.1109/ICICR61203.2024.00046.

[9] J. Qian, Z. Shang, S. Yan, Y. Wang, L. Chen, RoScript: a visual script driven truly non-intrusive robotic testing system for touch screen applications, Association for Computing Machinery (2020), https://doi.org/10.1145/3377811.3380431.

[10] V. Mousavi, M. Varshosaz, F. Remondino, S. Pirasteh, J. Li, A Two-Step Descriptor-Based Keypoint Filtering Algorithm for Robust Image Matching, IEEE Transactions on Geoscience and Remote Sensing 60 (2022) 1-21, 10.1109/TGRS.2022.3188931.

[11] S. Baduge, S. Thilakarathna, J. Perera, M. Arashpour, P. Sharafi, B. Teodosio, A. Shringi, P. Mendis, Artificial intelligence and smart vision for building and construction 4.0: Machine and deep learning methods and applications, Automation in Construction 141 (2022) 104440, https://doi.org/10.1016/j.autcon.2022.104440.

[12] R. Khalil, N. Saeed, M. Masood, Y. Fard, M. Alouini, T. Al-Naffouri, Deep Learning in the Industrial Internet of Things: Potentials, Challenges, and Emerging Applications, IEEE Internet of Things Journal 8(14) (2021) 11016-11040, 10.1109/JIOT.2021.3051414.

[13] Y. Hu, Q. Jia, Y. Yao, Y. Lee, M. Lee, C. Wang, X. Zhou, R. Xie, F. Yu, Industrial Internet of Things Intelligence Empowering Smart Manufacturing: A Literature Review, IEEE Internet of Things Journal, 11(11) (2024) 19143-19167, 10.1109/JIOT.2024.3367692

[14] H. Ahmad, A. Rahimi, Deep learning methods for object detection in smart manufacturing: A survey, Journal of Manufacturing Systems 64 (2022) 181-196, https://doi.org/10.1016/j.jmsy.2022.06.011.

[15] R. Khanam, M. Hussain, R. Hill, P. Allen, A Comprehensive Review of Convolutional Neural Networks for Defect Detection in Industrial Applications, IEEE Access 12 (2024) 94250-94295, 10.1109/ACCESS. 2024.3425166.

[16] S. Jha, R. Babiceanu, Deep CNN-based visual defect detection: Survey of current literature, Computers in Industry 148 (2023) 0166-3615, https://doi.org/10.1016/j.compind.2023.103911.

[17] C. Li, Y. Chen, Y, Shang, A review of industrial big data for decision making in intelligent manufacturing, Engineering Science and Technology, an International Journal 29 (2022) 101021, https://doi.org/10.1016/j.jestch.2021.06.001.

[18] J. Wang, C. Xu, J. Zhang, R. Zhong, Big data analytics for intelligent manufacturing systems: A review, Journal of Manufacturing Systems 62 (2022) 738-752, https://doi.org/10.1016/j.jmsy.2021.03.005.

[19] C. G.Harris, M.Stephens, A combined corner and edge detector, Alvey vision conference 15 (1988) 10-5244, doi:10.5244/C.2.23.

[20] D. G. Lowe, Distinctive image features from scale-invariant keypoints, International journal of computer vision 60 (2004) 91-110, https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[21] K. Mikolajczyk,C. Schmid, Scale & Affine Invariant Interest Point Detectors. InternationalJournal of Computer Vision 60 (2004) 63–86, https://doi.org/10.1023/B:VISI.0000027790.02288.f2.

[22] E. Rosten, T. Drummond, Machine Learning for High-Speed Corner Detection. Computer Vision--ECCV 2006: 9th European Conference on Computer Vision (2006) 430-443, https://doi.org/10.1007/11744023_34.

[23] M. Calonder, V.Lepetit, C.Strecha, P. Fua, BRIEF: Binary Robust Independent Elementary Features. Computer Vision--ECCV 2010: 11th European Conference on Computer Vision (2010) 778-792, https://doi.org/10.1007/978-3-642-15561-1_56.

[24] M. Brown, R. Szeliski, S. Winder, Multi-image matching using multi-scale oriented patches, 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) 1 (2005) 510-517, doi:10.1109/CVPR.2005.235.

[25] P. F. Alcantarilla, A. Bartoli, A. J. Davision, KAZE Features, Computer Vision--ECCV 2012: 12th European Conference on Computer Vision (2012) 214-227, https://doi.org/10.1007/978-3-642-33783-3_16.

[26] Z. Gan, X. Zheng, Y. Song, et al. Screen-shooting watermarking algorithm based on Harris-SIFT feature regions. SIViP 18 (2024) 4647–4660, https://doi.org/10.1007/s11760-024-03102-7.

[27] H. Fang, W. Zhang, et al. Screen-Shooting Resilient Watermarking, IEEE Transactions on Information Forensics and Security 14 (2019) https://doi.org/1403-1418.10.1109/TIFS.2018.2878541.

[28] L. Li, R. Bai, et al. Screen-Shooting Resilient Watermarking Scheme via Learned Invariant Keypoints and QT, Sensors 21 (2021) 1424-8220, https://www.mdpi.com/1424-8220/21/19/6554.

[29] Y. Bai, L. Li, et al. Fast Frequency Domain Screen-Shooting Watermarking Algorithm Based on ORB Feature Points, Mathematics 11 (2023) 2227-7390, https://www.mdpi.com/2227-7390/11/7/1730.

[30] X. Han, T. Leung, et al. Matchnet: Unifying feature and metric learning for patch-based matching, Proceedings of the IEEE conference on computer vision and pattern recognition (2015) 3279-3286, doi:10.1109/CVPR/2015.7298948.

[31] S. Zagoruyko, N. Komodakis, Deep compare: A study on using convolutional neural networks to compare image patches, Computer Vision and Image Understanding 164 (2017) 38-55, https://doi.org/10.1016/j.cviu.2017.10.007.

[32] E. Simo-Serra, E. Trulls, et al. Discriminative learning of deep convolutional feature point descriptors, Proceedings of the IEEE international conference on computer vision (2015) 118-126, doi: 10.1109/ICCV.2015.22.

[33] Y. Tian, B. Fan, F. Wu, L2-net: Deep learning of discriminative patch descriptor in euclidean space, Proceedings of the IEEE conference on computer vision and pattern recognition (2017) 661-669, doi: 10.1109/CVPR/2017.649.

[34] M. Ester, H. Kriegel, et al. A density-based algorithm for discovering clusters in large spatial databases with noise, AAAI Press (1996) 226-231.

[35] X. Ding, X. Zhang, et al. RepVGG: Making VGG-style ConvNets Great Again, Computer Vision and Pattern Recognition (2021), https://doi.org/10.48550/arXiv.2101.03697.

[36] Q. Hou, D. Zhou, J. Feng, Coordinate Attention for Efficient Mobile Network Design, Computer Vision and Pattern Recognition (2021), https://doi.org/10.48550/arXiv.2103.02907.

[37] D. Hendrycks, K. Gimpel, Gaussian Error Linear Units (GELUs), Machine Learning (2016), https://doi.org/10.48550/arXiv.1606.08415.

[38] D. DeTone, T. Malisiewicz, A. Rabinovich, SuperPoint: Self-Supervised Interest Point Detection and Description, Computer Vision and Pattern Recognition (2018), https://doi.org/10.48550/arXiv.1712.07629.

[39] P. Sarlin, D. DeTone, T. Malisiewicz, A. Rabinovich, SuperGlue: Learning Feature Matching with Graph Neural Networks, Computer Vision and Pattern Recognition (2020), https://doi.org/10.48550/arXiv. 1911.11763.

[40] J. Sun, Z. Shen, Y. Wang, H. Bao, X. Zhou, LoFTR: Detector-Free Local Feature Matching with Transformers, Computer Vision and Pattern Recognition (2021), https://doi.org/10.48550/arXiv.2104.00680.