

Parallel Algorithms to Detect and Classify Defects in Surface Steel Strips



Khaled R. Ahmed, Majed Al-Saeed, and Maryam I. Al-Jumah

1 Introduction

Worldwide, steel industry is one of the most important strategic industries. Quality is an important competitive factor to the steel industry success. Detection of surface defects devotes a large percent of quality control process to satisfy the customer's need [1], and [2]. Defect detection and classification can be accomplished manually by human labor; however, it will be slow and subject to human-made errors and hazards. Therefore, automatic traditional-inspection systems were developed to detect various faults. These include eddy current testing, infrared detection, magnetic flux leakage detection, and laser detection. These methods are not able to detect all the faults, especially the tiny ones [3]. This motivates many researchers [4, 5] to develop computer vision systems capable of classifying and detecting defects in ceramic tiles [6], textile fabrics [7], and steel industries [8]. Achieving defect detection, localization, and classification in real time is one of the challenges in the steel production process. Therefore, the main aim of this chapter is to propose parallel algorithms to detect and classify patches, scratches, and scale defects in surface steel strips in real time.

The rest of this chapter is organized as follows. Section 2 reviews the related works. Section 3 illustrates the proposed algorithm. Section 4 discusses the experiment setup and results. Section 5 concludes this chapter.

K. R. Ahmed (✉)

School of Computing, Southern Illinois University, Carbondale, IL, USA

e-mail: kahmed@cs.siu.edu

M. Al-Saeed · M. I. Al-Jumah

King Faisal University, Hofuf, Saudi Arabia

e-mail: alsaeed@kfu.edu.sa

© Springer Nature Switzerland AG 2021

H. R. Arabnia et al. (eds.), *Advances in Artificial Intelligence and Applied Cognitive Computing*, Transactions on Computational Science and Computational Intelligence, https://doi.org/10.1007/978-3-030-70296-0_40

543

2 Related Work

Image processing plays a major role in the steel production industry to enhance the quality of the products. In the literatures, many image-processing algorithms have been proposed to detect various defects by features extraction techniques. A plenty of features have been used including color, texture, shape, geometry features, etc., for defect localization and type identification [9]. The common techniques used for feature extraction in steel images are categorized into four different approaches [10]. These approaches are statistical methods, structural algorithms, filtering methods, and model-based techniques as shown in Fig. 1a.

Statistical approaches usually used histogram curve properties to detect the defects such as histogram statistics, autocorrelation, local binary patterns, grey level co-occurrence matrices [11], and multivariate discriminant function [12]. Image processing and edge detection algorithms are the basic operations used in structural approaches. Due to various defects depicting similar edge information, it is hard to classify the defect types. Filter-based methods involve convolution with filter masks for computing energy or response of the filter. Filters can be applied in frequency domain [13], in spatial domain, or in combined spatial frequency domain [14]. Model-based approaches include fractals, random field models, autoregressive models, and the epitome model [10] to extract a model or a shape from images. Figure 2 lists methods utilized to detect two types of surface defects on steel strips.

There are many approaches to extract features in parallel. Lu et al. [15] proposed an adaptive pipeline parallel scheme for constant input workloads and implemented an efficient version for it based on variable input workloads; they speed up to 52.94% and 58.82% with only 3% performance loss. Also, Zhang et al. [16] proposed a model to generate gray level run length matrix (GLRLM) and extracts multiple features for many ROIs in parallel by using graphical processing unit

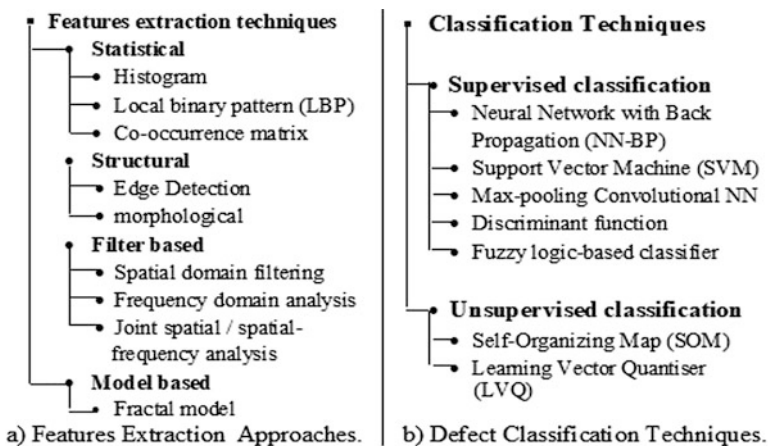


Fig. 1 Related work

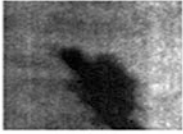

	 <p>Patches</p>	 <p>Scratches</p>
Detection Methods	<ul style="list-style-type: none"> • Local Binary Pattern (LBP) [3] • Histogram [24] • Edge Detection [24] • Features Extraction [10-26] • Algorithms [27] 	<ul style="list-style-type: none"> • Local Binary Pattern (LBP) [3] • Histogram [24] • Edge Detection [10] • Features Extraction [21] • Gray Level Co-occurrence Matrices (GLCM) [20-25]
Classification Methods	<ul style="list-style-type: none"> • Support Vector Machine (SVM) [3] • Nearest Neighbour Classifier (NNC) [3] • Deep Auto-encoder Network (DAN) [24] • Artificial Neural Networks (ANN) [24] • Learning Vector Quantization (LVQ) [30] 	<ul style="list-style-type: none"> • Nearest Neighbor Classifier (NNC) [3] • Support Vector Machine (SVM) [28] • Self-organizing Map (SOM) [18] • Gabor Filter [29] • Deep Auto-encoder Network (DAN) [24] • Artificial Neural Networks(ANN) [24]

Fig. 2 Defects detection and classification techniques

(GPU), and they achieved five-fold increase in speed than an improved sequential equivalent.

The classification process is the main consideration in the inspection system. Generally, there are two types of classification methods: supervised and unsupervised as presented in Fig. 1b. In supervised classification, training samples are labeled, and features are given to the classifier to generate the training model. The training model predicts the pre-defined classes for test samples [10]. These methods include SVM, neural networks, nearest neighbors, etc. Yazdchi et al. [17] applied neural network (NN) to classify steel images that achieved accuracy 97.9%. Yun et al. [18] suggested support vector machine (SVM) classifier for defect detection of scale-covered steel wire rods. In unsupervised classification, classifier earns on its own and it is not fed with labeled data. Classifier just tries to group together similar objects based on the similarity of the features [19]. Most common types of methods include K-means, SOM (self-organizing map) [20] and LVQ (learning vector organization) [13]. Figure 2 lists some defect detection and classification methods. The key parameters of the defect classification methods are the accuracy and the efficiency. This paper employs the SVM.

3 Proposed Algorithms

This chapter develops parallel algorithms to detect and classify patches, scratches, and scale defects in surface steel strip. Figure 3 shows the high-level design of the

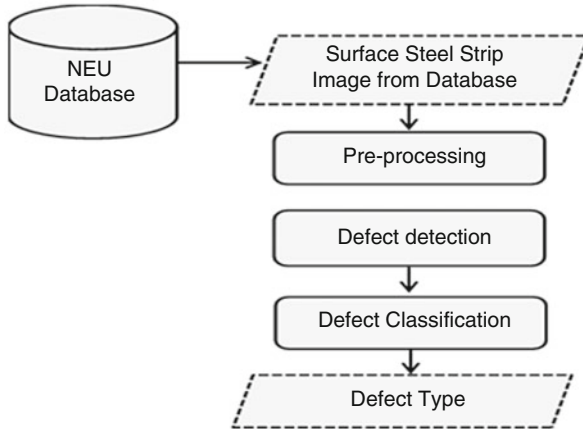


Fig. 3 High-level architecture of the proposed algorithm

proposed defect detection and classification technique. First phase is to preprocess the image to improve it and remove noises. Second phase detects defects from the steel image and segments it to defective ROIs. Third phase extracts *Haralick* features from gray level co-occurrence matrix (*GLCM*). Finally, these features will be used as inputs to the SVM classifier.

3.1 Preprocessing Phase

Surface steel images are subject to various types of noises due to image acquisition setup, lighting conditions, or material reflections. The preprocessing operation is an important step to eliminate light reflection and noises. Preprocessing operation carried out image enhancement and noise reduction. Image enhancement composes two steps to make image clearer. First, convert the RGB images into grayscale images and resize the image to $M \times N$. Then apply the contrast stretching operation to enhance image brightness by stretching the intensity values from 0 to 255. To remove noises, this chapter uses median filter to remove salt, pepper noises, and makes images more blurred [21, 22].

3.2 Defect Detection Phase

In this phase the algorithm divides the $M \times N$ grayscale steel image into blocks (ROIs) of size $W \times H$. After that, it extracts statistical features for each ROI by using multivariate discriminant function [12] to detect either the ROI is defected or not.

1. *Features Extraction*: The proposed algorithm divides the $M \times N$ grayscale image into ROIs of size $W \times H$, where $W \ll M$ and $H \ll N$. To characterize the shape of the surface defects and detect either if the ROI is defected or not, the algorithm extracts following statistical features for each ROI: difference (δ), mean (μ) and variance (ν) as in Eqs. (3), (4), and (5). After that, it calculates mean vector (MV) for each ROI as Eq. (6). Extract features need many operations that may take long time, which is not suitable to achieve real-time for defects detection. Consequently, this paper uses Summed Area Table (SAT) [3] to reduce the required time to compute these features. It quickly generates the sum of values of a rectangular subset of a grid using Eq. (1). Where $i(x, y)$ is the pixel value from the given image and $S(x, y)$ is the value of the summed area table [23]. For $M \times N$ image, SAT table is created with $O(M \times N)$ complexity. Once it is created, the task to calculate the sum of pixels in a rectangle that is a subset of the original image can be done in constant time by Eq. (2) with $O(1)$.

$$S(x, y) = i(x, y) + S(x-1, y) + S(x, y-1) - S(x-1, y-1) \quad (1)$$

$$\begin{aligned} \text{SUM} = & S(x_0-1, y_0-1) + S(x_0 + x_1, y_0 + y_1) \\ & - S(x_0 + x_1, y_0-1) - S(x_0-1, y_0 + y_1) \end{aligned} \quad (2)$$

$$\text{Diff_Value}(ROI, W, H) \quad (3)$$

$$\mu = \text{Mean_SAT}(Image, x_0, y_0, W, H) \quad (4)$$

$$\nu = \text{Variance_SAT}(Image, x_0, y_0, W, H) \quad (5)$$

$$\text{MV} = [\delta \ \mu \ \nu]^T \quad (6)$$

Consequently, SAT can quickly iterate pixels and significantly reduces the required time to process the images. In this paper, we developed SAT algorithm in parallel using CUDA [24, 25] as shown in Fig. 4.

2. *Defect Detection*: The defect detection algorithm divides image into ROIs to detect each ROI either belongs to defective group (G_1) or non-defective group (G_2). MV_1 and MV_2 are mean vectors that contain the statistical features of G_1 and G_2 respectively. We assume MV_{ROI} denotes a mean vector that contains the features in ROI [12]. The two groups represent defective pixels and non-defective pixels in the image. To separate the pixels into defective and non-defective pixels, we create two Gaussian Mixture Models (GMM)s [26]. An iterative Expectation-Maximization (EM) algorithm is used to estimate maximum likelihood ξ_1 and

```

Algorithm: Parallel SAT Function
1  function PSAT (image)
2  {
3      double *SAT;
4
5      //Allocate space on device
6      cudaMalloc(&SAT, M * N * sizeof(double));
7
8      //Copy matrix to device memory
9      cudaMemcpy(SAT, image, M * N * sizeof(double), cudaMemcpyHostToDevice);
10
11     // invoke kernel at host side
12     const int maxThreadsPerBlock = IMG_SIZE; // number of threads in each block
13     int numBlocksX = ((unsigned int) (M / maxThreadsPerBlock) + 1);
14     int numBlocksY = ((unsigned int) (N / maxThreadsPerBlock) + 1);
15
16     dim3 numBlocksForRows = dim3(numBlocksX, 1);
17     dim3 numBlocksForColumns = dim3(numBlocksY, 1);
18
19     //Run rows sum kernel
20     rowSum <<<numBlocksForRows, maxThreadsPerBlock >>>((double (*) [IMG_SIZE])SAT, M, N);
21     cudaDeviceSynchronize();
22
23     //Run columns sum kernel
24     colSum<<<numBlocksForColumns,maxThreadsPerBlock>>>((double (*) [IMG_SIZE])SAT, M, N);
25     cudaDeviceSynchronize();
26
27     //Copy data to host memory
28     cErr = cudaMemcpy(image, SAT, M*N * sizeof(double), cudaMemcpyDeviceToHost);
29
30     // free device global memory
31     cudaFree(d_Data);
32 }

```

```

Algorithm: Sum Rows Kernel
1  function sumRows (SAT, M, N)
2  {
3      int idx = blockIdx.x * blockDim.x + threadIdx.x;
4      if (idx < N)
5          for (int i = 0; i < M; i++)
6              SAT[i][idx] = SAT[i][idx] + SAT [i - 1][idx];
7  }

```

```

Algorithm: Sum Columns Kernel
1  function sumCols (SAT, M, N)
2  {
3      int idx = blockDim.x * blockIdx.x + threadIdx.x;
4      if (idx < M)

```

Fig. 4 Parallel SAT Algorithm in CUDA

\mathcal{L}_2 for both GMM_1 and GMM_2 as in Eq. (7), by guess weight α , mean m and variance σ values [27]. EM contains three steps. First step chooses initial parameters values, the second is E-step that evaluates the responsibilities using the current parameter values, the third is M-step that re-estimates the parameters using the current responsibilities [28]. By maximum likelihood function $ML(p)$ in Eq. (8), the pixel belongs to G_1 if \mathcal{L}_1 is larger than or equal to \mathcal{L}_2 otherwise it belongs to G_2 Eq. (8).

$$\mathbf{t} = \log(\alpha) + (2\pi\sigma^2)^{-\frac{1}{2}} \exp\{-\frac{1}{2}((x-m)/\sigma)^2\} \tag{7}$$

$$ML(p) = \begin{cases} p \in G_1, & \mathbf{t}_1 \geq \mathbf{t}_2 \\ p \in G_2, & \mathbf{t}_1 < \mathbf{t}_2 \end{cases} \tag{8}$$

To decide if the image is defective or not, we apply multivariate discriminant function, Ω , for each ROI in the image [5, 12]. Multivariate discriminant function applies *Mahalanobis* distance rule Δ^2 Eq. (9) [12, 29]. If *Mahalanobis* distance between the ROI and G_1 more than or equal *Mahalanobis* distance between ROI and G_2 , then the ROI is defective; otherwise, the ROI is non-defective as in Eq. (10). Multivariate discriminant function in Eq. (11) derived from Eqs. (9) and (10), where T denotes matrix transpose [12]:

$$\Delta^2 (MVROI, MV) = (MVROI - MV)^T CV^{-1} (MVROI - MV) \tag{9}$$

$$\Delta^2 (MVROI, MV_1) \geq \Delta^2 (MVROI, MV_2) \tag{10}$$

$$\Omega = (MV_1 - MV_2)^T CCV^{-1} MVROI - 1/2(MV_1 - MV_2)^T CCV^{-1} (MV_1 - MV_2) \tag{11}$$

To apply discriminant function Ω , we need to calculate covariance vector, CV , for both groups G_1 and G_2 by Eq. (12), where N_i denotes the number of pixels in the group and x_{ij} denotes pixel in G_1 and G_2 . Then the common covariance matrix (CCV) will be calculated by Eq. (13):

$$CV_i = \frac{1}{N_i - 1} \sum_{j=1}^{N_i} (x_{ij} - MV_i) (x_{ij} - MV_i)^T \text{ for } i = 1, 2. \tag{12}$$

$$CCV = \sum_{j=1}^2 (N_j - 1) \frac{CV_j}{n - 2} \text{ where } n = \sum_{j=1}^2 n_j \tag{13}$$

The ROI is defective if the value of discriminant function Ω is positive. Otherwise, the ROI is non-defective as Eq. (14). To decide either the image contains defects or not, it must have at least one defective ROI; otherwise, the image is non-defective [12].

$$\Omega (\text{block}) = \begin{cases} \text{defective block}, & \Omega \geq 0 \\ \text{non - defective block}, & \Omega < 0 \end{cases} \tag{14}$$

Applying the discriminant rule, Ω , for all ROIs in the image, results would be like in Fig. 5; the numbers represent the value of the discriminant rule for each ROI [12]. The image has no defect if all ROIs have negative discriminant value.

To speed up the EM algorithm, this chapter calculates each iteration E-step and M-step for all pixels in parallel using CUDA [30] as shown in Fig. 6. The parallel EM algorithm has main function $PEM()$ that launches the GPU kernel $UpdateKernel()$ to process E-step and M-step for all pixels in parallel as seen in Fig. 7. The $UpdateKernel()$ creates 1D grid and 1D ROIs; each ROI contains MT threads [25]. Each thread calculates both E-step and M-step for a pixel. Assume an image has NP pixels, the complexity of the sequential EM is $O(\text{Maximum of Iteration} \times NP)$. However, it is $O(\text{Maximum of Iteration})$ for the proposed parallel EM.

3.3 Defect Classification

In the past decade, different researchers have presented several methods for steel defect classification [31]. Nevertheless, these methods are limited to high computation and low accuracy. This work proposed a classification algorithm to classify scratch, patches, and scale defects. The algorithm has two modules. First, features extraction module takes the defective image and calculates GLCM and Haralick features [32, 33]. Second, once features are extracted, the classification module utilizes support vector machine (SVM) for the recognition of their corresponding class.

1. *Features Extraction Module*: GLCM defines the texture of an image by calculating how frequently pairs of pixels with specific values and in a specified spatial relationship happen in an image. Each element (i, j) of GLCM denotes how many times the gray levels i and j occur as a sequence of two pixels located at a defined distance δ along a chosen direction θ . *Haralick* defined a set of 14 measures of textural features [33]. This work selected six textural features shown in Table 1

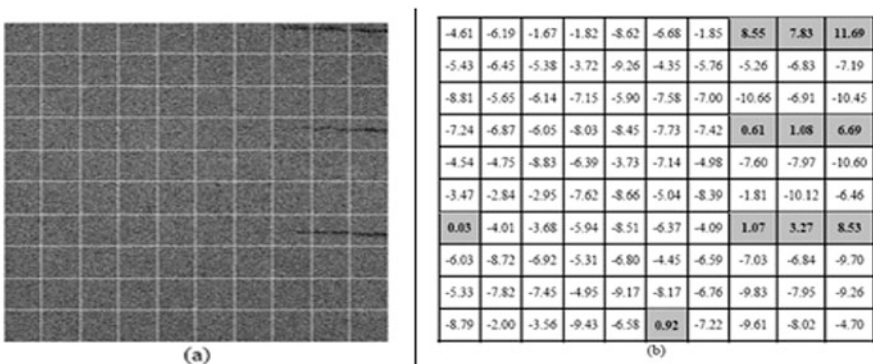


Fig. 5 Defective image and its discriminant result. [12]

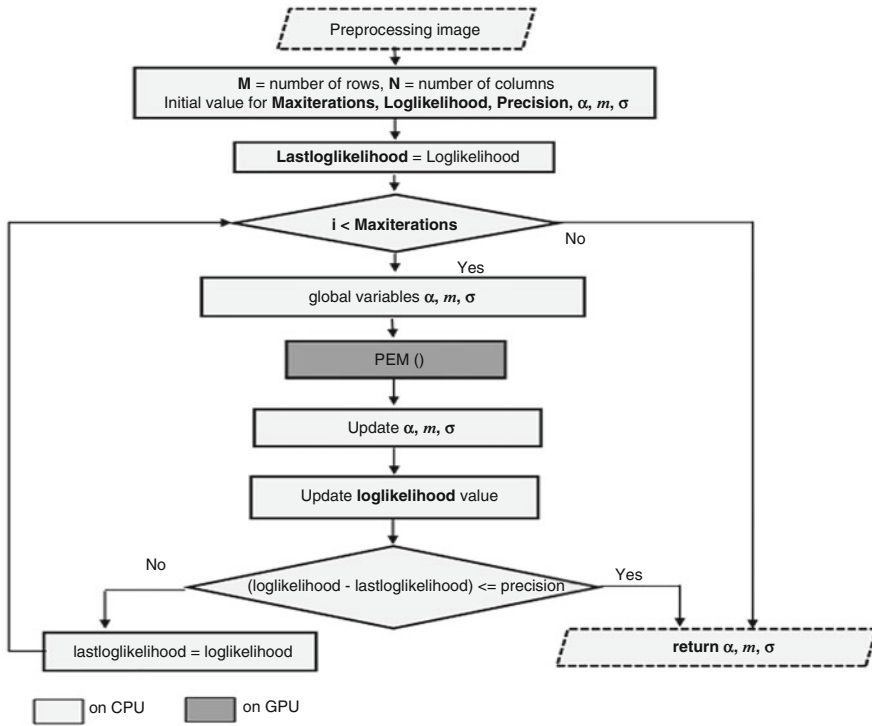


Fig. 6 Parallel EM algorithm

that are used as input to SVM classifier to classify the defect in steel image. The computation time of texture features depends on the number of gray levels, between 0 and 255 levels. This chapter develops *Haralick* features calculation in parallel to reduce the execution time for the proposed classification algorithms [34]. To extract the features from GLCM matrix in parallel, this work developed *P_Haralick_Features()* function that launches the *HaralickKernel()* kernel with 2D (blocks) ROIs with n_x threads in the x-direction and n_y threads in the y-direction [25]. Each thread computes six features for one pixel. To accumulate features values from threads, kernel uses *AtomicAdd()* function as shown in Fig. 8. While a thread reads a memory address, it adds the value of feature to it, and writes back the result to memory. As GLCM is 256×256 matrix, 256 gray levels then the complexity to extract *Haralick* features by sequential algorithm is $O(256 \times 256)$; however, with parallel algorithm it is $O(1)$.

2. *Defect Classification Module*: To classify surface steel strip defects, this chapter uses multi-class classification SVM. The classification process is divided into two steps: training process and testing process. In both steps the classifier will use features vectors; in the training process to label different defects and in the test

```

Algorithm: Update Kernel called by the PEM()
1  function UpdateKernel(double *x, double *dev_sum_wj, double *dev_sum_
   *a, double *mean, double *var, double *dev_L, const int numGaussians, co
2  {
3      int idx = blockIdx.x * blockDim.x + threadIdx.x;
4      int j = idx;
5
6      double resp[2];
7
8      if (j < dataSize)
9          double den;
10         double l_max, tmp, sum;
11         l_max = -1000000;
12         for (int i = 0; i < numGaussians; i++)
13             resp[i] = log(a[i]) + Probability(x[j], mean[i], var[i]);
14             if (resp[i] > l_max) l_max = resp[i];
15             sum = 0;
16         for (int i = 0; i < numGaussians; i++)
17             sum += exp(resp[i] - l_max);
18         tmp = l_max + log(sum);
19         dev_L[j] = tmp;
20
21         den = 0;
22         for (int i = 0; i < numGaussians; i++)
23             resp[i] = exp(resp[i] - tmp);
24             den += resp[i];
25
26         double *learn_a = &dev_sum_wj[j] * numGaussians;
27         double *learn_m = &dev_sum_wj_xj[j] * numGaussians;
28         double *learn_v = &dev_sum_wj_xj2[j] * numGaussians;
29
30         for (int k = 0; k < numGaussians; k++)
31             learn_a[k] = resp[k] / den;
32             learn_m[k] = x[j] * resp[k] / den;
33             learn_v[k] = x[j] * x[j] * resp[k] / den;
   }

```

Fig. 7 UpdateKernel() invoked by EM algorithm in CUDA

Table 1 Haralick features

Homogeneity	$F1 = \sum_m \sum_n (m - n)^2 \text{GLMC}(m, n)$
Entropy	$F2 = \sum_m \sum_n \frac{1}{1+(m-n)^2} \text{GLMC}(m, n)$
Energy	$F3 = \sum_m \sum_n \text{GLMC}(m, n) \log \text{GLMC}(m, n)$
Mean	$F4 = \sqrt{\sum_m \sum_n \text{GLMC}^2(m, n)}$
IDM	$F5 = \sum_m \sum_n \frac{1}{2} (m \text{GLMC}(m, n) n \text{GLMC}(m, n))$

step to classify defects [2, 18]. This work extracts features in parallel to reduce the classification time. In the training phase, we pass these features along with their corresponding labels to the SVM to generate SVM model. The second and final step of the proposed system is the testing phase where we have a test dataset

```

Algorithm: Extract Haralick Features Kernel
1 function HaralickKernel (GLMC, energy, contrast, homogeneity, IDM, entropy, mean)
2 {
3   int i = blockIdx.x * blockDim.x + threadIdx.x;
4   int j = blockIdx.y * blockDim.y + threadIdx.y;
5
6   if (i < HARLICK_SIZE && j < HARLICK_SIZE)
7     *energy = GLCM[i][j] * GLCM [i][j]; //Energy
8     atomicAdd(contrast, (i - j)*(i - j)* GLCM[i][j]); //Contrast
9     atomicAdd(homogeneity, GLCM [i][j] / (1 + abs(i - j))); //Homogeneity
10    if (i != j)
11      atomicAdd(IDM, GLCM[i][j] / ((i - j)*(i - j))); //IDM
12    if (data[i][j] != 0)
13      atomicAdd(entropy, -1 * GLCM[i][j] * log10(GLCM[i][j])); //Entropy
14    atomicAdd(mean, 0.5*(i* GLCM[i][j] + j * GLCM[i][j])); //Mean
15 }
    
```

Fig. 8 *HaralickKernel* () to extract *Haralick* features

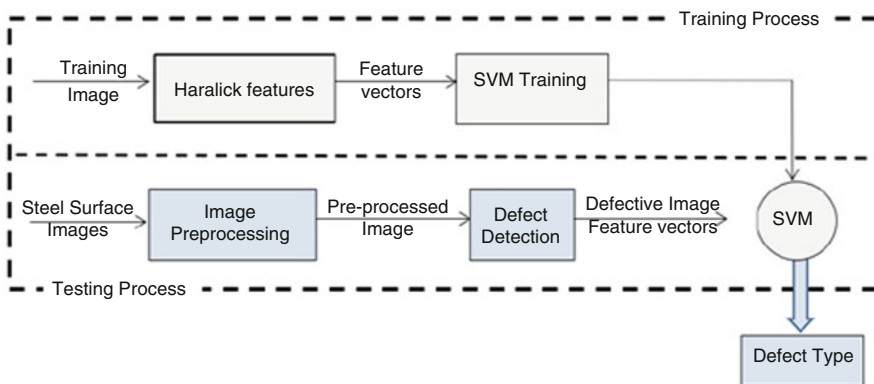


Fig. 9 Defect classification steps

images of the steel strips. These images are further checked for the defect; if an image has defective ROIs, then Haralick features must be extracted. These features are then given to the SVM along with a trained SVM model which was trained in the first step; as a result, SVM identifies the predicted class of defect. Figure 9 shows the classification steps.

3.4 Evaluation Criteria of Defect Detection and Classification

This section introduces the performance criteria to check the effectiveness and accuracies of the defect detection and classification algorithms.

1. *Detection Accuracy*: The defect detection accuracy as shown in Eq. (15) is used to determine the accuracy and the effectiveness of the defect detection algorithms [35, 36]:

$$DA = \frac{TP + TN}{TP + TN + FP + FN} \quad (15)$$

where TN is true negative, TP is true positive, FN is false negative, and FP is false positive. True positive is referred to defective steel image identified as defective. True negative is referred to defect-free steel image identified as defect-free. False positive is referred to defect-free steel image identified as defective. False negative is referred to defective steel image identifies as defect-free.

Classification Accuracy: The accuracy of the classification algorithm could be calculated as in Eq. (16):

$$\text{Accuracy}(d_j) = \frac{N_c(d_j)}{N_t(d_j)} \quad (16)$$

where d_j is the defect class j , $j = 1, \dots, W$, $N_c(d_j)$ is the number of images correctly classified as defect class d_j , $N_t(d_j)$ is the total number of images in that defect class, and W is the total number of defected classes. The total accuracy for the defect classification algorithm is the probability of a correct prediction due to our classifier for all defect classes over a set of images:

$$\text{Total accuracy} = \frac{\sum_j^W \text{Accuracy}(d_j)}{W} \quad (17)$$

2. *Performance Criteria*: Computing time is the main criteria to study the performance of the proposed defect detection and classification algorithm. The required time to detect and classify defects for steel surface is divided into two main significant parts: detection time and classification time as shown in Eq. (18):

$$\text{TotalTime} = \sum_{i=1}^B Dt_i + Ct_j \quad (18)$$

where the surface steel image has been divided into B ROIs, Dt_i is the required time to detect either ROI i in the surface steel image has defect or not, Ct_i is the required

time to classify the type of the defects in the defected ROI i in the surface steel image. Ct_i equals zero if ROI i has no defects. In addition, this work used speedup to measure the performance of the proposed algorithm is speedup as in Eq. (19):

$$\text{Speedup} = \frac{T_s}{T_p} \quad (19)$$

where T_s denotes the execution time of the sequential algorithms, and T_p denotes the execution time of parallel algorithms.

4 Experiment Results

This section introduces experiments results of the proposed parallel algorithms for detecting scratch, patches, and scale defects.

1. *Setup*: The experiment platform in this work is Intel(R) Core™ i7-8550U with a clock rate of 1.8 GHz, working with 8 GB DDR4 RAM and a graphics card that is NVIDIA GeForce 940MX with 2GB of DDR3 and 1122 MHz. All experiments in this project were conducted in Windows 10 64-bit operating system with the development environments of Visual C++ 2017 with *OpenCV* 3.4.3 and *CUDA* toolkit 10. NEU dataset has 1800 grayscale steel images has been used. It includes six types of defect which are inclusion, crazing, patches, pitted surface and rolled-in scale, 300 samples for each type. Moreover, to study the tolerance of the proposed algorithm against noises this paper added salt and pepper noises to about 1%–2% of the steel images dataset. Dataset is divided into 70% for training set and 30% for testing set.
2. *Experiment*: The experiments were conducted in three stages: pre-processing, defect detection, and defect classification. In the first stage, images are pre-processed as follows. Steel images are resized to 400×400 and then a 3×3 median filter is used to remove noises. The second stage “defect detection” includes four steps. The first step creates two Gaussian mixture models for each image by maximum likelihood to divide the image pixels into two groups: defective group and non-defective group. Figure 10 shows two GMM for steel image having scratch defect. The second step calculates statistical features mean, difference, and variance for these groups. In third step, each image is divided into ROIs. Each ROI contains 40×40 pixels. Use summed area table to extract statistical features for each ROI. Finally, use the discriminant rule to decide either the ROI is defective or non-defective. The fourth step displays defected ROIs if the steel image is defective or not as shown in Fig. 11. The defect classification stage is divided into two phases. In the training phase, the SVM classifier takes vectors of the extracted six *Haralick* features with associated labels for all images in the training set and then generates a training model. In the testing phase, the

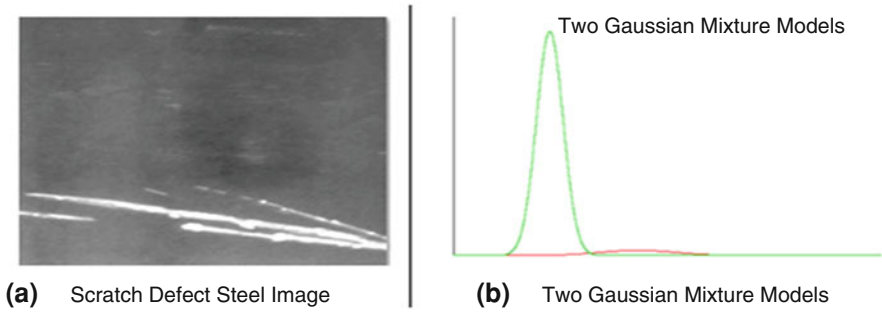
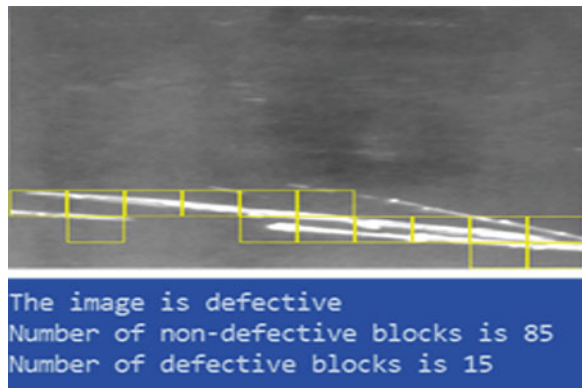


Fig. 10 Steel image and GMM models

Fig. 11 Defect detection result

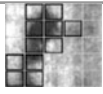
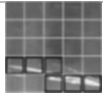
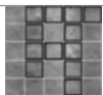


trained SMV takes *Haralick* features as a vector for test image from testing set and predicts defect class.

3. *Results*: This section illustrates gradually the results of the proposed algorithms. In this chapter we develop three defect detection algorithms: (1) sequential without SAT algorithm (SEQ) [12], (2) proposed sequential with SAT algorithm (SSAT), and (3) proposed parallel with SAT algorithms (PSAT) developed by CUDA. The median execution time for three types of implementations to detect and classify three defects will be illustrated in this section. Table 2 shows the median defect and classification time in milliseconds (ms) for SEQ and SSAT and PSAT algorithms. They detect three defects, patches, scratch, and scale, while image size is 400×400 pixels and ROIs size is 40×40 pixels. Table 2 contains steel images with defected ROIs, defect type, median of the execution time for defect detection and classification algorithms, and speedup. The rightmost column in this table displays the speedup of the PSAT compared to SEQ.

Figure 12 shows median execution time for sequential and parallel algorithms implemented to detect and classify three defects. It depicts that the *PSAT* algorithm is the fastest one especially in detecting scratch defect. The *PSAT* algorithm is able to accomplish $\sim 1.50x$ speedup. Figure 13 plots median defect detection and

Table 2 Execution times for three algorithms

Steel image	Defect type	Median execution time in ms			Speedup
		SSAT	SEQ	PSAT	
	Patches	22.045	20.094	14.656	1.50
	Scratch	22.787	21.697	13.804	1.65
	Scale	20.319	20.015	14.577	1.39

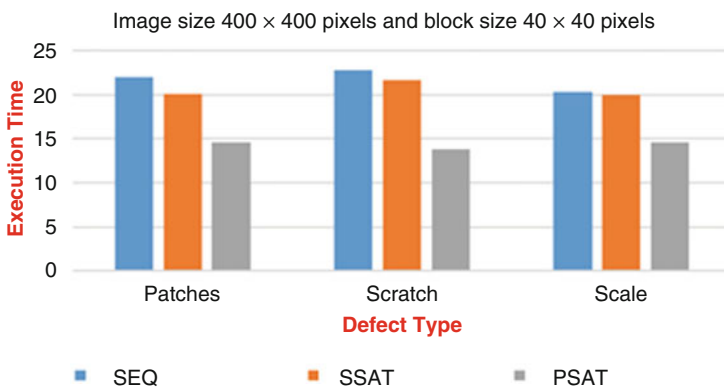


Fig. 12 Median defect detection and classification time

classification time with different dimensions of surface steel images with block (ROI) size 40×40 pixels. It shows that the median execution increases linearly with the increase of image size. The proposed PSAT algorithm has exhibited superior performance compared to the other algorithms while image size is increasing.

The proposed algorithm divides the image into non-overlapped ROIs (partitions). The number of ROIs is specified based on defect location. Some defect may split into two ROIs. So, the smaller defect in a ROI may not be classified as a defect type. In doubt, this case will affect the accuracy of the proposed algorithm. The number of ROIs must be chosen carefully to reduce the defect splitting. Figure 14 depicts that PSAT algorithm takes shortest execution time in milliseconds for all block sizes, while the SEQ takes significantly long execution time. SEQ divides image into ROIs with size $W \times H$ and handles each ROI separately, while PSAT generates 2D ROIs with $W \times H$ threads. Each thread launches kernel to detect either ROI is defective or not in parallel. Therefore, PSAT shows 1.4 speedup compared with SSAT and more than 1.6 speedup compared with SEQ. The accuracy of the proposed defect detection algorithms SSAT and PSAT is about 95.66%.

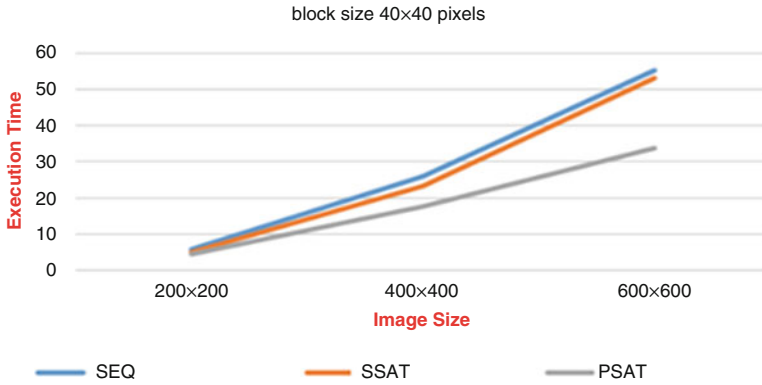


Fig. 13 Median defect detection and classification time with image size

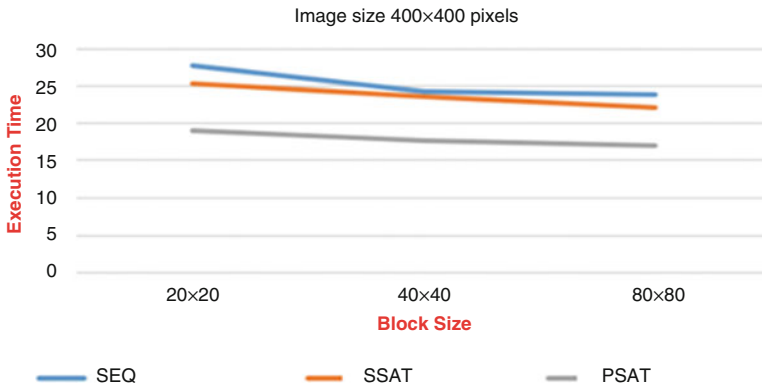


Fig. 14 Median defect detection and classification time with ROI size

5 Conclusion

The major aim of this chapter is to design and develop a parallel algorithm that automates the defects inspection process in steel industry. This work employed SAT to improve the defect detection algorithm in [12]. In addition, it demonstrated the detailed implementation of the proposed sequential algorithm based on SAT and parallel algorithm. Once defected image is detected, SVM classifier has been used to classify the type of the defect (scratch, patches, scale). The experimental results in this article verified that the developed techniques succeeded to speed up the surface steel defects detection and classification compared with the existing techniques. Finally, the proposed parallel algorithm speeds up over the sequential algorithms developed in [12] by about 1.65 times to detect scratch, about 1.5 times to detect patch and 1.39 times to detect scale defects respectively where the image size is 400×400 with about 95.66%.

References

1. Mostafa Sadeghi, Hossein Soltani, Kamran Zamanifar, Application of parallel algorithm in image processing of steel surfaces for defect detection, Special Issue: Technological Advances of Engineering Sciences, Fen Bilimleri Dergisi (CFD), Cumhuriyet University, Turkey. 36(4), 263-273 (2015)
2. K. Song, Y. Yunhui, A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects. Appl. Surf. Sci. **285**, 858–864 (2013)
3. S. Tian, X. Ke, An algorithm for surface defect identification of steel plates based on genetic algorithm and extreme learning machine. Metals **7**(8), 311 (2017)
4. R. Khaled, Fast and parallel summed area table for fabric defect detection. Int. J. Pattern Recognit. Artif. Intell. **30**(09), 1660004 (2016)
5. N. Neogi et al., Review of vision-based steel surface inspection systems. EURASIP J. Image Video Process. **2014**(1), 50 (2014)
6. R. Khaled, A. Nahed, An efficient defect classification algorithm for ceramic tiles. *IEEE 13th Int. Symp. on Autonomous Decentralized System (ISADS)*, pp. 255–261 (2017)
7. H. Sager, E. Loay, George. Defect detection in fabric images using fractal dimension approach. *International Workshop on Advanced Image Technology*, vol. 2011 (2011)
8. S. Zhou et al., Classification of surface defects on steel sheet using convolutional neural networks. *Materiali Tehnologije* **51**(1), 123–131 (2017)
9. T. Ramesh, B. Yashoda, Detection and Classification of Metal Defects using Digital Image Processing, pp. 31–36 (2014)
10. Xianghua Xie, A review of recent advances in surface defect detection using texture analysis techniques, *Electronic Letters on Computer Vision and Image Analysis*, 7(3), 1-22 2008. Computer Vision Center / Universitat Autònoma de Barcelona, Barcelona, Spain
11. D. Wang, et al., Wood surface quality detection and classification using gray level and texture features. *Int. Symp on Neural Networks*. Springer, Cham (2015)
12. L. Weiwei, et al., Automated on-line fast detection for surface defect of steel strip based on multivariate discriminant function. *Intelligent Information Technology Application, IEEE. IITA'08. Second Int. Symp on. Vol. 2* (2008)
13. G. Wu, et al., A bran-new feature extraction method and its application to surface defect recognition of hot rolled strips. *IEEE Int. Conf. on Automation and Logistics* (2007)
14. Y. Zhang et al., Fabric defect detection and classification using gabor filters and gaussian mixture model, in *Asian Conference on Computer Vision*, (Springer, Berlin, Heidelberg, 2009)
15. Y. Lu, Li, et al., Parallelizing image feature extraction algorithms on multi-core platforms. *J. Parallel Distrib. Comput.* **92**, 1–14 (2016)
16. H. Zhang et al., GPU-accelerated GLRLM algorithm for feature extraction of MRI. *Sci. Rep.* **9**(1), 1–13 (2019)
17. M. Yazdchi, et al., Steel surface defect detection using texture segmentation based on multifractal dimension. *Int. Conf. on. IEEE Digital Image Processing* (2009)
18. J.P. Yun, et al., Vertical scratch detection algorithm for high-speed scale-covered steel BIC (Bar in Coil). *Int. Conf. on. IEEE Control Automation and Systems (ICCAS)* (2010)
19. P. Caleb, M. Steuer, Classification of surface defects on hot rolled steel using adaptive learning methods. *Knowledge-Based Intelligent Engineering Systems and Allied Technologies, IEEE Proc. Fourth Int. Conf. on. Vol. 1* (2000)
20. T. Maenpaa, Surface quality assessment with advanced texture analysis techniques. *Proc. of Int. Surface Inspection Summit*, Luxembourg (2006)
21. S.R. Mahakale, N.V. Thakur, A comparative study of image filtering on various noisy pixels. *International Journal of Image Processing and Vision Sciences* **1**(2), 69–77 (2012)
22. I. Singh, N. Neeru, Performance comparison of various image Denoising filters under spatial domain. *International Journal of Computer Applications* **96**(19), 21–30 (2014)
23. F.C. Crow, Summed-area tables for texture mapping. *ACM SIGGRAPH Comput. Graph.* **18**(3), 207–212 (1984)

24. D. Kirk, NVIDIA CUDA software and GPU parallel computing architecture. Proceedings of the 6th International Symposium on Memory Management, ISMM 2007, Montreal, Quebec, Canada, (2007), pp. 103-104. DOI: <https://doi.org/10.1145/1296907.1296909>
25. C. Zeller, *Cuda c/c++ Basics* (NVIDIA Coporation, NVIDIA, Santa Clara, California, USA, 2011)
26. H. Bensmail et al., Regularized Gaussian discriminant analysis through eigenvalue decomposition. *J. Am. Stat. Assoc.* **91**(436), 1743–1748 (1996)
27. B.W. Silverman, On the estimation of a probability density function by the maximum penalized likelihood method. *Annals of Statistics* **10**(3), 795–810 (1982). <https://doi.org/10.1214/AOS/1176345872>
28. L. Machlica, Fast estimation of Gaussian mixture model parameters on GPU using CUDA. *12th IEEE Int. Conf. on Parallel and Distributed Computing, Applications and Technologies* (2011)
29. G.J. McLachlan, Mahalanobis distance. *Resonance* **4**(6), 20–26 (1999)
30. G. Noaje, et al., Source-to-source code translator: OpenMP C to CUDA. *IEEE 13th Int. Conf. High Performance Computing and Communications (HPCC)* (2011)
31. C. Park, W. SangChul, An automated web surface inspection for hot wire rod using undecimated wavelet transform and support vector machine. *Industrial Electronics, Annual 35th Conference of IEEE, IECON'09* (2009)
32. P. Mohanaiah, P. Sathyanarayana, L. GuruKumar, Image texture feature extraction using GLCM approach. *International Journal of Scientific and Research Publications* **3**(5), 290–294 (2013)
33. R.M. Harlick et al., Textural features for image classification. *IEEE Trans. Syst. Man Cybernet.* **SMC-3**(6), 610–621 (1973)
34. A. Parvez, C.P. Anuradha, Efficient implementation of GLCM based texture feature computation using CUDA platform. *Int. Conf. on Trends in Electronics and Informatics (ICEI)* (2017)
35. M. Arun, R. Prathipa, P.S.G. Krishna, Automatic defect detection of steel products using supervised classifier, *International Journal of Innovative Research in Computer and Communication Engineering, India.* **2**(3), 3630-3635 (2014).
36. R. Mishra, D. Shukla, A survey on various defect detection. *Int. J. Eng. Trends Technol.* **10**(13), 642–648 (2014)