electronics

Article

A Smart and Robust Automatic Inspection of Printed Labels Using an Image Hashing Technique

Mehshan Ahmed Khan [1], Fawad Ahmed [2], Muhammad Danial Khan [3], Jawad Ahmad [4,*], Harish Kumar [5] and Nikolaos Pitropakis [4]

[1] Department of Electrical Engineering, HITEC University, Taxila 47080, Pakistan; 16-ms-ee-009@hitecuni.edu.pk
[2] Department of Cyber Security, Pakistan Navy Engineering College, National University of Sciences and Technology, Karachi 75350, Pakistan; fawad@pnec.nust.edu.pk
[3] Department of Machine Learning Automation, CONVSYS(Pvt.) Ltd., Islamabad 44020, Pakistan; muhammad.danialkhan@convsys.net
[4] School of Computing, Edinburgh Napier University, Edinburgh EH10 5DT, UK; n.pitropakis@napier.ac.uk
[5] Department of Computer Science, College of Computer Science, King Khalid University, Abha 61413, Saudi Arabia; hrangaiah@kku.edu.sa
* Corresponding: j.ahmad@napier.ac.uk

Abstract: This work is focused on the development of a smart and automatic inspection system for printed labels. This is a challenging problem to solve since the collected labels are typically subjected to a variety of geometric and non-geometric distortions. Even though these distortions do not affect the content of a label, they have a substantial impact on the pixel value of the label image. Second, the faulty area may be extremely small as compared to the overall size of the labelling system. A further necessity is the ability to locate and isolate faults. To overcome this issue, a robust image hashing approach for the detection of erroneous labels has been developed. Image hashing techniques are generally used in image authentication, social event detection and image copy detection. Most of the image hashing methods are computationally extensive and also misjudge the images processed through the geometric transformation. In this paper, we present a novel idea to detect the faults in labels by incorporating image hashing along with the traditional computer vision algorithms to reduce the processing time. It is possible to apply Speeded Up Robust Features (SURF) to acquire alignment parameters so that the scheme is resistant to geometric and other distortions. The statistical mean is employed to generate the hash value. Even though this feature is quite simple, it has been found to be extremely effective in terms of computing complexity and the precision with which faults are detected, as proven by the experimental findings. Experimental results show that the proposed technique achieved an accuracy of 90.12%.

Keywords: image hashing; fault detection; speeded up robust features (SURF); robustness and detection; maximum likelihood estimation sample consensus (MLESAC); feature matching; region of interest (ROI) extraction; ROI matching; feature extraction time; hash matching

Citation: Khan, M.A.; Ahmed, F.; Khan, M.D.; Ahmad, J.; Kumar, H.; Pitropakis, N. A Smart and Robust Automatic Inspection of Printed Labels Using an Image Hashing Technique. *Electronics* **2022**, *11*, 955. https://doi.org/10.3390/electronics11060955

Academic Editor: Chiman Kwan

Received: 26 January 2022
Accepted: 15 March 2022
Published: 19 March 2022

1. Introduction

Labels printed on products are usually made up of paper, metal, plastic film, cloth, etc. A label contains information about a product and hence it is necessary that products contain clearly printed labels. Inspection of labels ensures that labels are printed correctly and placed at the proper position. Inspection can be performed manually or automatically. Manual inspection requires a person to check each label, which slows down the process. This method is rarely used today because the results of an inspection vary from person to person. To overcome this problem, sophisticated computer vision algorithms are used. Label inspection is performed by mounting single or multiple cameras on an assembly line as shown in Figure 1. The camera captures images containing labels one at a time and

sends them to a computer, where a computer vision algorithm checks each label for any possible fault. If the label is tampered due to a printing defect, then a feedback signal is sent to the assembly line to reject the label. Despite the fact that automatic inspection is accurate and faster than manual inspection, there are many problems that make it difficult to design an automatic label inspection algorithm. For example, the captured image may undergo distortions due to changes in appearance, illumination, colour, shape, rigid transformation and non-rigid transformations. Moreover, it is a challenging task to recognize objects in images with complex backgrounds depending upon the viewpoint [1].

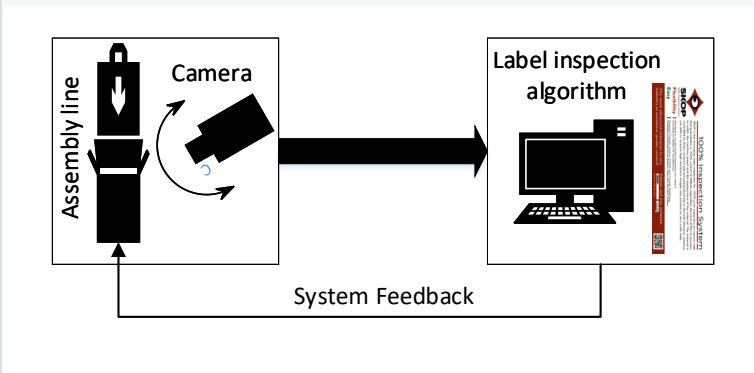


Figure 1. Automatic label inspection on an assembly line.

In this paper, a new technique is proposed using image hashing to detect printing faults in product labels. The hash of an image gives a compact and fixed-length numeric representation of image content and has been widely used for integrity verification [2–6]. The advantage of using image hashing is its robustness to content preserving operations, like compression, contrast enhancement, geometric transformations, etc. On the other hand, the image hash should be sensitive in detecting tampering and can further locate tampered areas. It is imperative to note that traditional cryptographic hash functions such as MD5, SHA1, SHA256, etc., are not suitable for label inspection application because they are sensitive to a single bit change in pixel value, and hence are not robust to content preserving operations [7,8]. To design an image hashing algorithm, both robustness and discrimination are the main factors. There is a trade-off between these two properties, which means that if robustness is increased then discrimination is reduced and vice versa [9,10].

For label inspection, a hashing algorithm should be fast and highly robust to geometric transformations and sensitive to detect minute level tampering. It is difficult to find all these properties in a single hashing scheme. In this paper, a new image hashing scheme is proposed to verify the integrity of printed labels. The proposed scheme is resilient to content preserving operations like geometric transformations and is able to detect minute level tampering. Therefore, any small area which has a printing fault can be easily detected. Secondly, due to the robustness property of image hashing, there is a very low false rejection of correctly printed labels. Generally, the false rejection occurs due to alignment, lightning and scaling issues. Most of the image hashing techniques presented so far mainly focus on multimedia security and image authentication purposes. In our work, we extended the application of image hashing and used it for the label inspection work. Label inspection requires that the algorithm be fast and detect faults at high accuracy. To make the inspection process fast we used a simple yet effective approach to detect tampering in labels. Experimental results show that the proposed scheme can detect faults in labels at high speed with high accuracy. In Section 2, a review of a number of image hashing schemes proposed in the literature is presented. The hashing methodology adopted in this paper for label inspection is illustrated in Section 3. The proposed hashing scheme for label inspection is presented in Section 4. Experimental results are given in Sections 5 and 6 concludes the paper.

2. Related Work

Conventionally, perceptual image hashing is used in multimedia security, content authentication and image forensics. In recent years, many researchers have used image hashing in various applications. For example, to search an image in large databases [11], facial recognition [12] and image authentication [13]. In the last decade, many perceptual image hashing techniques have been developed to detect tampering in images. In early techniques, Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT) were used to create image hash. Fridrich et al. [14] used DCT coefficients to extract image content and proposed a hashing scheme that is used in the application of digital watermarking. Venkatesan et al. [15] constructed image hash by using statistics of DWT coefficients. This approach makes the scheme robust against geometric distortions and compression. Karsh et al. [16] presented a hashing scheme which uses Discrete Wavelet Transform-Singular Value Decomposition (DWT-SVD) and the spectral residual method for hash formulation. From an image, DWT-SVD extract global features and the spectral residual model extract the saliency regions. In another work, Karsh et al. [17] used global and local features of an image obtained by using ring partitions and salient regions for hash construction. Qin et al. [18] utilized the Dual Cross Pattern (DCP) and salient structural features. In this scheme, a Gaussian low pass filter and Singular Value Decomposition (SVD) operation were applied on the image. Texture features were extracted from DCP and all the salient features were combined to compress the textual features to generate a robust image hash. Experiments show that the proposed scheme gives better performance in anti-collision attacks.

Tang et al. [19] proposed a hashing scheme based on Colour Vector Angles (CVA). For the calculation of CVA, the input image was first normalized by interpolation and low pass filtering. A histogram was extracted within an inscribed circle of the normalized image. Finally, a hash was formed by compressing the histogram. Tang et al. [20] extended his work and applied DCT, DWT and Discrete Fourier Transform (DFT) on the histogram to obtain a hash of small size. In both these techniques, the calculation of CVA takes all the pixels into account which makes the scheme resistant to rotation. In a recent work, Tang et al. [21] proposed rotation invariant image hashing. From the normalized image, the rotation invariant feature matrix was extracted through Log Polar Transform (LPT) and DFT. By using Multi-Dimensional Scaling (MDS), the algorithm learns a discriminative and compact representation of the feature matrix.

Ouyang et al. [22] presented a new hashing scheme that uses combined Scale-Invariant Feature Transform (SIFT) and quaternion Zernike moments to extract features. Only the top k features are selected to construct image hash and repeated key points were removed during feature extraction. This method can locate tampering and also classifies the type of tampering in an image. The hash produced in this scheme is short and the hashing is robust to large angle rotation. Vadlamudi et al. [23] used the n distinct SIFT feature points from the Lightness (L) component of $L^*a^*b^*$ (L. Lightness, a. red-green and b. yellow-blue) colour image space to extract the image content. DWT is subsequently applied to the extracted content to obtain approximation coefficients. Binary hash is formed by normalizing the approximation coefficients thus making the proposed method invariant to many distortions, for example, compression, scaling, filtering, contrast adjustment and brightness.

Gharde et al. [9] introduced a dual perceptual hash function to generate a hash using a fuzzy colour histogram. The $L^*a^*b^*$ colour space was used in this scheme. An unbiased fuzzy colour histogram was generated through tuning factor, which improves the system robustness and discriminative capability. Ng et al. [24] utilized Convolutional Neural Networks (CNN)s to develop the scheme named Multi-Level Supervised Hashing (MLSH) with deep features. The multiple-hash-table mechanism was integrated into the mechanism to extract the features from deep convolutional layers of the network so that both the semantic and structural information can be preserved. Moreover, the recall and precision rate of the scheme illustrates the better performance of the algorithm than existing techniques. Hosny et al. [7] introduced a hashing scheme which is based on

Quaternion Polar Complex Exponential Transform (QPCET). In this scheme, the input image is first normalized using bicubic interpolation and the Gaussian low pass filter is then applied. This is followed by hash construction from features that are extracted through QPCET moments. Pun et al. [25] proposed a feature extraction method that combines local features of both structure and colour information. The hash is constructed through Horizontal Location-Context Hashing (HLCH) and Vertical Location-Context Hashing (VLCH) methods. Image authentication and tamper localization methods are also proposed. The results in [25] show robustness against different signal processing and geometric attacks. Fei et al. [26] proposed a hash-based template matching algorithm for video sequences. To improve tracking, Laplace-based Hash (LHash) and Laplace-based Difference Hash (LDHash) are proposed. In this scheme, video objects are tracked efficiently as compared to conventional tracking algorithms such as mean-shift or Compressive Tracking (CT) with low-computational cost. Ji et al. [27] proposed an image segmentation algorithm for Synthetic Aperture Radar (SAR) images which is based on perceptual image hashing. First, segmentation is done with multi-thresholding followed by removal of speckle noise and transforming the image in the DCT domain. The DCT coefficients are then compressed by using Principle Component Analysis. Hash values of the image regions are obtained by applying the Different Hashing Algorithm [28]. To achieve better results, the uneven background is removed with morphological methods.

The image hashing techniques reviewed above, and, generally speaking, nearly all the hashing techniques proposed in the literature are robust to a limited type of non-malicious distortion. It is difficult to find a single image hashing scheme that is resilient to all types of non-malicious distortions. Secondly, some of the techniques are slow and cannot be applied in real time. In image hashing, the type of features selected has a significant impact on the performance of the hashing scheme. The fundamental requirement of fault detection in printed labels is that the hashing scheme should be fast and resilient to geometric transformations and distortions due to illumination and noise. To address these issues, the statistical mean of non-overlapping image blocks is used as a feature to generate the image hash. This feature, although very simple, requires less computational time and has been found to be robust. To undo geometric distortions, SURF features are used for the alignment of query and template images. This makes the proposed scheme invariant to geometric and other distortions.

3. Methodology

The general block diagram of the proposed hashing algorithm used for label inspection is shown in Figure 2. The whole process consists of two parts; in the first part, hash and SURF features of the template image is obtained. In the second part, before calculating the hash of the query image, the Region of Interest (ROI) containing the label under consideration is extracted. This is done so that one-to-one matching between the template label image and the query label image becomes possible. The process requires SURF features of both template and the query image. To find tampered areas in the query image, it is matched with the hash of the template image.

To list all possible faults, a database of actual labels supplied by Sprint Digital Sdn Bhd and SKOP Malaysia is used in this paper. Images in the database were collected from an assembly line by installing a camera to capture the printed labels. In this database, some images are faulty and some are without any fault. At first, possible sets of faults are identified manually by observing the labels. The proposed hashing algorithm then identifies the labels as correct or faulty. From the database, one image is selected as the template image in which there is no fault, as shown in Figure 3. All the images are compared with the template image using perceptual image hashing to identify any possible tampering.

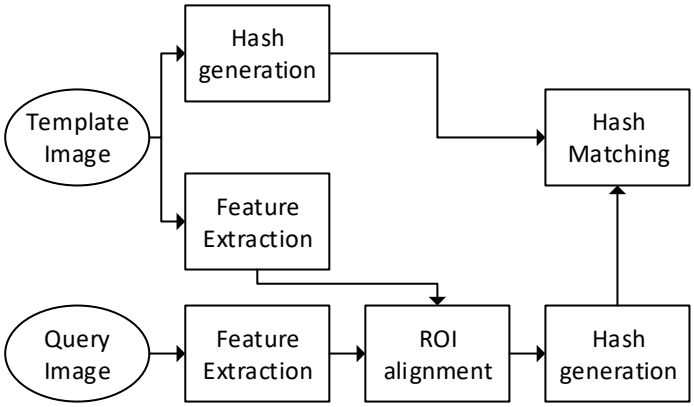


Figure 2. Flow diagram of label inspection using perceptual image hashing.

Figure 3. Template image.

There are two major issues that need to be resolved before calculating the hash of the label under consideration. If these two problems are not addressed, then a correct label could be classified as incorrect. The first problem is the partial presence of other labels in the captured frame as shown in Figure 4a–g. Interestingly, these seven images also have printing faults that are encircled. The hashing algorithm should ignore the partial presence of other labels and only detect fault(s) in the ROI under consideration which is shown by the bounding box. The second problem is due to geometric transformation between the template image and the query image labels as shown in Figure 5. The hashing algorithm should be robust or invariant to geometric transformations. In the proposed hashing algorithm, SURF features are used to undo geometric distortions.

To match a template label image with the given query image label, there should be a correspondence between them. Many algorithms have been proposed to find correspondence between images. These algorithms have been used in a wide a range of applications like object tracking [29], image fusion [30], image registration [31], and object detection and recognition [32,33]. To extract the ROI containing the label under consideration, the ROI of the template image is required. The goal is to detect the template ROI in the query image. To solve this problem, a reliable and fast ROI extraction method is required. Various approaches have been proposed to recognize features in a digital image, for example, colour histograms [34], receptive field histograms [35], eigenspace matching [36], etc. When an image is subjected to affine transformation, rotation, scaling and or illumination, most of these detection algorithms may not be robust against all types of distortions. Moreover, they are comparatively slow and cannot be used in real-time applications which require algorithms to be fast. In 1999, Lowe proposed SIFT [37] to detect features that are invariant to a number of distortions. SIFT, however, is not fast enough for real-time applications. To ensure high-speed feature detection, the Speeded-Up Robust Feature (SURF) detector was introduced in 2006 [38]. The SURF algorithm is able to detect features in an image similar to SIFT, but at a higher speed because of low dimensionality feature descriptors.

(**a**)

(**b**)

(**c**)

(**d**)

(**e**)

(**f**)

(**g**)

Figure 4. Images collected from assembly line with faults highlighted. (**a**) Tampered image with minor area misprinted; (**b**) Tampered image with text on bottom misprinted; (**c**) Tamper image with letter 'S' misprinted and black colored QR code on the bottom has red patched; (**d**) Tampered image with large portion of the label missing; (**e**) Tampered image minor line size area missing; (**f**) Tampered image with few letters from the description has missing; (**g**) Tampered image with small patch area missing.

(**a**)

(**b**)

(**c**)

Figure 5. Images with background and marked ROI. (**a**) ROI is on the centre but left label has more area than right; (**b**) ROI is on the centre but right label has slight more area than left; (**c**) ROI is on the right side but on the left major portion of upcoming label is also present.

Feature extraction produces a "descriptor" which is a vector with numerical values. In the feature extraction stage of SURF, descriptors are computed around the interest points, also called key points. Hence, a descriptor is defined as a numerical feature obtained around a key point. Traditionally, these descriptors are matched by using Euclidean distance. This method, however, produces a lot of outliers, thus significantly reducing the accuracy of matching. If matching is inaccurate, then the geometric transformation between template and query images will not be correct. This will directly affect the ROI extraction. To filter out these outliers and correctly estimate geometric transformation between the template and label images, SURF features are used along with Maximum Likelihood Estimation Sample Consensus (MLESAC) [39] algorithm to match images. SURF features are chosen because experimental results discussed in Section 5.2 show that it gives robust and fast results when compared with other known feature extraction algorithms like KAZE [40], MSER [41] and BRISK [42]. By using MLESAC, maximization of log-likelihood estimates is transformed into minimization of the cost function [39]. This method effectively removes mismatched feature points and easily extract the ROI which contains the label under consideration.

4. The Proposed Label Inspection Algorithm

In this section, a new scheme is proposed to automatically inspect printed labels using SURF features for alignment of images and perceptual image hashing for image matching. Figure 6 shows the detailed block diagram of the proposed scheme along with the different types of algorithms used in each step. The entire process consists of three parts as explained below.

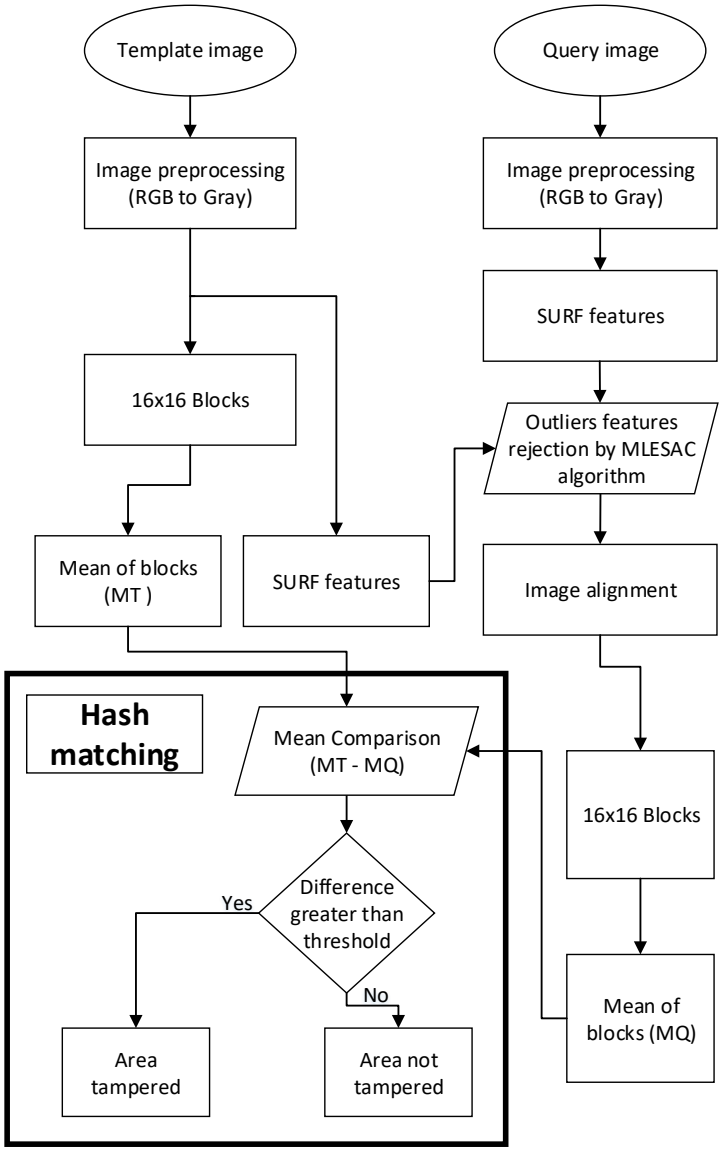


Figure 6. Block diagram of the proposed hashing algorithm for label inspection.

1. Template image hash: The image which is selected as a template should be error-free because the hash of other images is compared with this image. First, the template image is converted into grey scale to reduce processing time. To calculate the hash of the template image, it is divided into 16×16 non-overlapping blocks and the mean value of each block is calculated. Mean values of all the blocks are finally concatenated to generate the hash of the template image. Both SURF features and the image hash are stored in a database. SURF features are used for ROI alignment of the query label image in the image matching module.
2. Query image hash: To calculate the hash of the query image, it is first aligned with the template image because images captured from the camera are generally misaligned as shown in Figure 5. To properly align query and template images, SURF features of both template and query images are used by the MLESAC algorithm to extract ROI from the query image. By finding the ROI, it is ensured that the query image does not contain extra patches of proceeding or preceding labels. The extracted ROI is divided into 16×16 non-overlapping blocks and the mean of all the blocks are concatenated to obtain the hash of the query image.

3. Hash matching: To compare the hash of both template and query images, the mean value of each block of the template image is matched with the corresponding block mean of the query image. The sum of the absolute difference between template and query image hash is compared with the chosen threshold to decide whether the corresponding area is tampered or not. In case of a printing fault in the label, the difference between the mean value of the corresponding spatial area will be greater than the threshold. This enables not only the detection of the faulty area but also localizing its exact spatial location where the printing error has occurred in the label.

Following are the main steps of the proposed hashing algorithm.

1. Image pre-processing.
2. Image alignment.
3. ROI extraction.
4. Hash generation.
5. Hash matching.

These steps are summarized in Algorithm 1 and details of each step are presented in the following sections.

Algorithm 1: Hash generation steps.

Template image hash
1. Read the template image.
2. Convert the image from RGB to grey scale.
3. Extract SURF feature vector for the template image, F_s.
4. Divide the image into 16×16 blocks.
5. Find mean of the image blocks, M_T.
6. F_s and M_T are stored in the database.
Hash of query image
1. Read the query image.
2. Convert the image from RGB to grey scale.
3. Extract SURF feature vector of the query image F_q.
4. Outliers rejection and image alignment using MLESAC algorithm using F_s and F_q
5. Divide the aligned image into 16×16 blocks.
6. Find the mean of the image blocks M_Q.
7. M_Q is stored in the database.
Hash matching
1. $M_{diff} = M_T - M_Q$
if $M_{diff} > th$
area of the image is tampered
elseif $M_{diff} \leq th$
area of the image is not tampered

4.1. Image Pre-Processing

In this step, the RGB label image is converted to grey scale by using National Television Standards Committee (NTSC) technique [1] as follows:

$$g(x,y) = 0.2989 * I(x,y,R) + 0.5870 * I(x,y,G) + 0.1140 * I(x,y,B) \quad (1)$$

In Equation (1), $g(x,y)$ is the grey scale image and $I(x,y,R)$, $I(x,y,G)$ and $I(x,y,B)$ are the red, green and blue pixel values of the RGB image, respectively. Grey scale conversion is performed to reduce the processing time of the hashing algorithm. Figure 7 shows the grey scale version of both template and query images.

(**a**)

(**b**)

Figure 7. Grey scale version of template and query images. (**a**) Template image; (**b**) Query image.

4.2. Image Alignment

Image alignment is the most important part of the proposed scheme. The captured query image may undergo various geometric transformations like translation, rotation, scaling or their combination. Hence it is necessary that before calculating the hash of the query image, the query image should first be aligned with the template image. The alignment parameters are obtained by matching the SURF features of the query image and the template image. Once the query image is aligned with the template image, the ROI from the query image is extracted by cropping with respect to the coordinates of the template image. The process of alignment consists of the following steps:

4.2.1. Feature Detection

The amount of feature detection varies from image to image because each image is captured differently. Figure 8 shows SURF [38] feature detection for template and query images.

(**a**)

(**b**)

Figure 8. SURF detected features. (**a**) SURF features of the template image; (**b**) SURF features of the query image.

4.2.2. Feature Matching

Feature matching finds correspondence between the template and query images. The process of matching is based on the sum of absolute differences. Some factors like image size and image quality affect the time and reliability of feature matching. For a visual representation, the matched features points of both images detected in the previous step are joined by lines as shown in Figure 9.

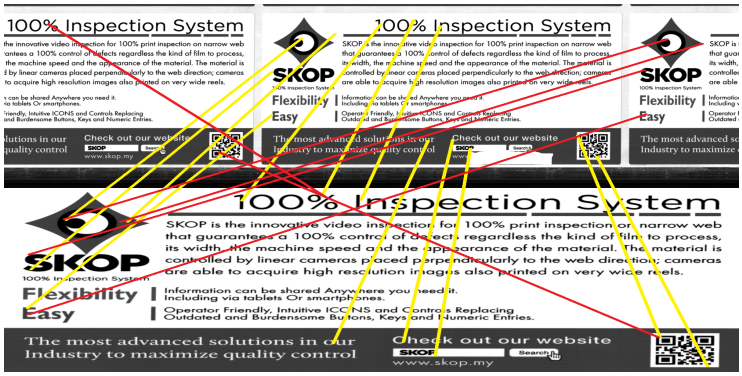


Figure 9. Feature matching.

4.2.3. Outliers Rejection and Region of Interest (ROI) Alignment

Outliers are those features that are generated due to false interest points. In Figure 9, it can be clearly seen that there are a lot of false points which are indicated by red lines, while inliers are shown by yellow lines. The outliers are mainly due to camera orientation because whenever a camera captures an image, it usually contains background along with the ROI. For example, portions of other labels marked by a square can be seen in Figure 10.

Figure 10. Sample images captured by the camera.

Due to possible geometric transformation, the ROI of the query image may not be properly aligned with the template, as shown in Figures 8 and 9. In addition, portions of proceeding and preceding labels are also captured in the frame. To correctly match template and query images, the ROI of the query image should be properly aligned so that

the exact portion of the misprinted area could be located. To address this problem, the image captured by the camera is modelled as a rotated, scaled and translated version of the template image. The transformation matrix of such an image is given as follows [1]:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = [T] \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{2}$$

$$X' = TX. \tag{3}$$

Let X' be the resultant image after applying transformation matrix T on the image X.

$$X' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}, X = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

where T is given by

$$T = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}. \tag{4}$$

The matrix T can be any transformation matrix such as translation (T_r), scaling (S) or rotation (R). These transformations can be expressed as [1]:

$$T_r = \begin{bmatrix} 1 & 0 & t_1 \\ 0 & 1 & t_2 \\ 0 & 0 & 1 \end{bmatrix}. \tag{5}$$

$$R = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{6}$$

$$S = \begin{bmatrix} S_1 & 0 & 0 \\ 0 & S_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{7}$$

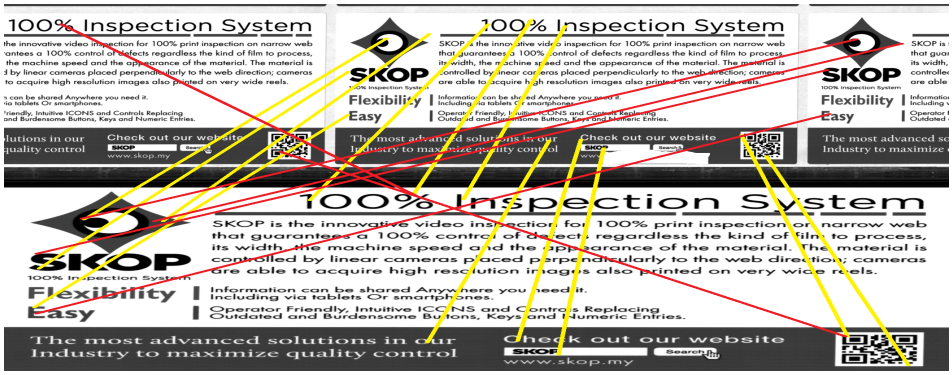
By combining the transformations given by Equations (5)–(7), new transformations can be obtained as shown below.

$$E = \begin{bmatrix} R & T_r \end{bmatrix} \quad S_i = \begin{bmatrix} SR & T_r \end{bmatrix} \quad A_f = (S, T_r, S_i)$$

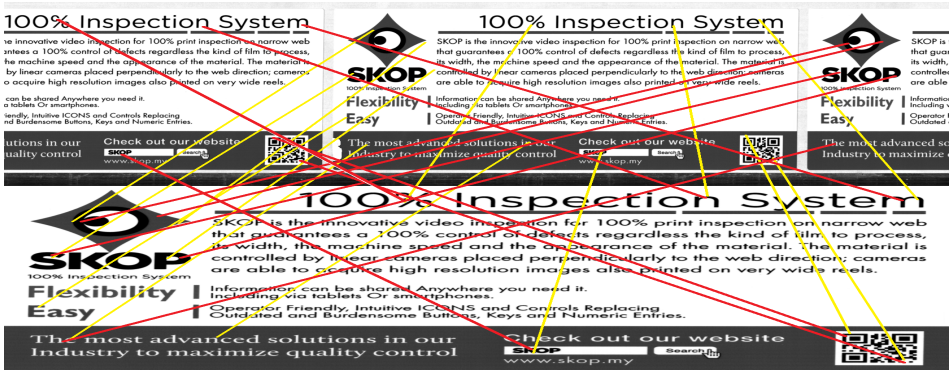
For example, the Euclidean (E) transformation is a combination of translation and rotation. The similarity (S_i) transformation is a combination of scaling, translation and rotation, while the affine (A_f) transformation is a combination of scaling, rotation, translation and similarity transformations.

To overcome the problems due to outliers rejection and image alignment, the RANSAC algorithm can be used but the Maximum Likelihood Estimation Sample and Consensus (MLESAC) algorithm gives better performance than RANSAC [39]. Both RANSAC and MLESAC are used when input data is contaminated with outliers. When the MLESAC algorithm is applied, the data is divided into inliers and outliers. It uses feature vectors from both the template and query images and approximate the transformation matrix to project the query image on the template image and eliminate all the outliers points [39].

In Figure 11a,b, feature matching results on sample images indicate that all the outliers are not rejected, due to which the remaining outliers interfere with the results of matching. Figure 11c,d are the results obtained after applying the MLESAC algorithm.



(**a**)



(**b**)

(**c**)

(**d**)

Figure 11. Feature matching on sample images. (**a**) Feature point matching with outliers (Example 1); (**b**) Feature point matching with outliers (Example 2); (**c**) Outliers rejected by the MLESAC algorithm (Example 1); (**d**) Outliers rejected by the MLESAC algorithm (Example 2).

4.3. ROI Extraction

The MLESAC algorithm gives the output transformation matrix, T_q which is used to extract the ROI from the query image.

$$X_q = T_q X_{qq} \tag{8}$$

In Equation (8), X_q is the aligned query image with portions of proceeding and preceding labels being cropped and X_{qq} is the actual query image. Therefore, the image X_q contains only the portion of the query image (ROI) which is to be matched with the template image. Results obtained after applying the MLESAC algorithm are shown in Figure 12. There is a large number of SURF feature points, however, a limited number of feature points are shown in Figure 11 for ease of illustration.

(**a**)

(**b**)

Figure 12. Aligned images. (**a**) Query image with small tampered area has been aligned with template image; (**b**) Query image with large tampered area has also been aligned with template image.

4.4. Hash Generation

The aligned images shown in Figure 12a is divided into 16×16 non-overlapping blocks. Let $B_1, B_2, \cdots, B_n$ be the blocks of image and $M_1, M_2, \cdots, M_n$ be the mean of each block. Let B_1 be the first block of the query image and $a_{1,1}, a_{1,2}, \cdots, a_{16,16}$ be the raw pixel values of B_1 as shown in Equation (9).

$$B_1 = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,16} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,16} \\ \vdots & \vdots & \ddots & \vdots \\ a_{16,1} & a_{16,2} & \cdots & a_{16,16} \end{bmatrix}, \tag{9}$$

The mean of B_1, represented as M_1 is calculated by using Equation (10).

$$M_1 = \frac{\sum_{i=1}^{16} a_{1,i}/16 + \sum_{i=1}^{16} a_{2,i}/16 + \cdots + \sum_{i=1}^{16} a_{16,i}/16}{16}. \quad (10)$$

In a similar way, mean of all the blocks are calculated. The hash of the template image (H_i^T) is formed by concatenating all the mean value of the blocks as shown by Equation (11).

$$H_i{}^T = [M_1{}^T, M_2{}^T, \cdots, M_n{}^T] \quad (11)$$

Similarly, the hash of the query image (H_i^Q) is expressed as:

$$H_i{}^Q = [M_1{}^Q, M_2{}^Q, \cdots, M_n{}^Q] \quad (12)$$

where $[M_1{}^T, M_2{}^T, \cdots, M_n{}^T]$ and $[M_1{}^Q, M_2{}^Q, \cdots, M_n{}^Q]$ in Equations (11) and (12) represents mean of all blocks in template and query image.

4.5. Hash Matching

The mean obtained in Section 4.4 is used to detect misprinted or tampered areas in the query image using hashing. To calculate the hash of the template image, the steps discussed in Section 4.1 are applied and the hash are generated by using Equation (10). The steps discussed in Section 4.2.1 is also applied to the template image to obtain SURF feature vectors, which are stored in a database. SURF features are used for the ROI alignment of the query image. To calculate the hash of the query image, all the steps discussed from Sections 4.1–4.4 are applied to the query image and this process also uses the stored SURF feature vectors of the template image which is already present in the database. The integrity of the query image is checked by calculating the hash distance between the template and the query image.

$$M_{di} = |H_i{}^T - H_i{}^Q| \quad (13)$$

In Equation (13), M_{di} is a matrix containing the sum of the absolute difference between the hashes of each block. The decision of which area is tampered is based on M_{di}. Ideally, if the area is not tampered, then its absolute difference will be zero, However, due to distortion like scaling, rotation etc., the value of M_{di} is slightly greater than zero. This problem can be solved by selecting a suitable threshold (Th); if M_{di} is greater than the predefined threshold it means that the block is tampered, otherwise not.

$$Decision = \begin{cases} Image\ block\ i\ is\ tampered & : M_{di} > Th \\ Image\ block\ i\ is\ not\ tampered & : M_{di} \leq Th \end{cases} \quad (14)$$

The threshold (Th) value should be selected in such a way that it should not be very large nor too small. If the selected threshold is too large, the hashing algorithm will not detect any tampered points even if they are present. Similarly, if the selected threshold is too small, then outliers will be detected and the algorithm may falsely detect them as tampered regions.

5. Experimental Results

In this section, the adopted methodology to experimentally evaluate parameters of the proposed scheme such as threshold selection, ROC curve and implementation time is presented.

5.1. Threshold Value Selection

To select a suitable threshold, the proposed algorithm is tested on different threshold values to obtain variation between threshold values and the accuracy of the system. The accuracy of the system is defined as:

$$Accuracy = \frac{Correctly\ identified\ images}{Total\ number\ of\ images} \tag{15}$$

To experimentally estimate the accuracy of the proposed hashing scheme, a labelled dataset of 81 images captured from a real assembly line are used from which 41 images are tampered. The results of applying the proposed hashing scheme to all the images are shown in Figure 13. It is observed that by increasing the threshold, the accuracy of the system increases at the start but when the threshold value reaches around 65, the accuracy starts to decrease because the algorithm identifies all the 81 images as correct. After doing a number of experiments, the threshold value of 65 has been found to give good results with an accuracy of 90.12%.

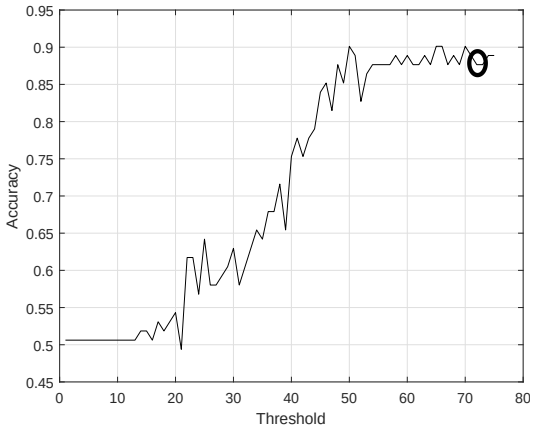


Figure 13. Plot of the system threshold vs. accuracy.

To identify the spatial position area in which tampering has been detected, a binary mask of the same size as that of the query image is generated. If a particular block is tampered, then it is displayed in white and all other blocks are displayed as black. Figure 14a,c,e,g,i,k,m shows labels that are captured from the assembly line and misprinted areas are highlighted by a square. Figure 14b,d,f,h,j,l,n shows the output of the proposed label inspection algorithm which detects the misprinted regions displayed as white. The black portion indicates that the area is not tampered. For example, Figure 15b is fully black, which indicates that the image is not tampered. Although the algorithm works on grey scale images, it can also detect printing of wrong colours. This happens because the change in colour changes the grey scale value which affects the mean of the corresponding block. For example, in Figure 14e, there are two faults in the image; the letter '*S*' is missing and the colours are misprinted, both of these faults were successfully detected by the proposed algorithm.

Figure 14. Tampered labels detected by the proposed algorithm. (**a**) Query image has minor area misprinted; (**b**) White area on the bottom left indicates that the query image has fault on the corresponding bottom area; (**c**) Query image with text and small patch on the bottom misprinted; (**d**) Binary mask has white area which is in same shape as that of misprinted area in query image; (**e**) Query image has misprinted fault in logo; (**f**) Arrow shaped white area in the binary mask indicates that the query image has arrow shaped fault. shape; (**g**) Query image has major portion missing; (**h**) Patterns on the binary mask indicates that the query image has missed logo and some text; (**i**) Query image has minor line sized area missing; (**j**) Binary mask detected the line shape fault in the query image; (**k**) Query image missed some text from the description; (**l**) Binary mask identifies the location of missing text; (**m**) Query image has small patch size area missing; (**n**) Binary mask identifies the location of missing patch.

(**a**) (**b**)

Figure 15. Correct label. (**a**) Query image with no fault; (**b**) Fully black mask indicates that the query image has no fault.

5.2. Implementation Time

In the proposed scheme, SURF feature extraction and alignment consumes most of the computation time. There are many other algorithms, for example, KAZE [40], MSER [41] and BRISK [42] which give similar performance as SURF but they are slow compared to SURF. To compare the processing time of SURF with other algorithms, an experiment was performed on a laptop with an Intel Core i7 (4500U) processor, 8 GB RAM and MATLAB 2018a. Table 1 gives an idea about the timing of the proposed algorithm when SURF features are replaced with other features. The timing of SURF comes out to be the fastest, i.e., 5.07 s among all the other algorithms.

Table 1. Time for different features.

Feature	Computational Time (Seconds)
MSER	52.24 s
KAZE	19.97 s
BRISK	10.32 s
SURF	5.07 s

A high processing time of 5.07 s is probably due to the presence of high-resolution images. Although, algorithm has the capability to process images of any given size but most of the images have sizes of around $3391 \times 1017 \times 3$. However, this much time is not acceptable for real-time applications. To overcome this problem, the proposed algorithm was also implemented in the Python programming language using the Open-CV library which helped to reduce the implementation time from 5.07 s to 0.4 s.

5.3. Receiver Operating Characteristics (ROC) Curves

To evaluate the performance of the proposed scheme and its performance between discrimination and robustness, parameters such False Acceptance Rate (*FAR*) and False Rejection Rate (*FRR*) are computed. *FAR* is defined as the number of times tampered images are detected as authentic. *FRR* is defined as the number of times authentic images are detected as tampered. *FAR* and *FRR* are defined as [13].

$$FAR = \frac{n_1}{N_1} \tag{16}$$

$$FRR = \frac{n_2}{N_2} \tag{17}$$

In Equation (16), n_1 is the number of tampered images that are misclassified as genuine and N_1 is the total number of images in the dataset. Similarly, in Equation (17), n_2 is the number of genuine images which are misclassified as tampered and N_2 is the total number of genuine images in the dataset.

To estimate *FAR* and *FRR*, all the images in the dataset are used. One image is selected as the template and the hash of all the 81 images are compared with it. The threshold is varied from 1 to 75 to obtain *FAR* and *FRR* values. Both *FAR* and *FRR*, as shown in Figures 16 and 17, are constant in the start due to the fact that the algorithm parameters

are not optimized to work on less threshold. When the threshold value starts increasing, the FAR starts increasing and FRR starts decreasing. After reaching the threshold value of 65, both FAR and FRR become constant and therefore 65 is selected as a suitable threshold.

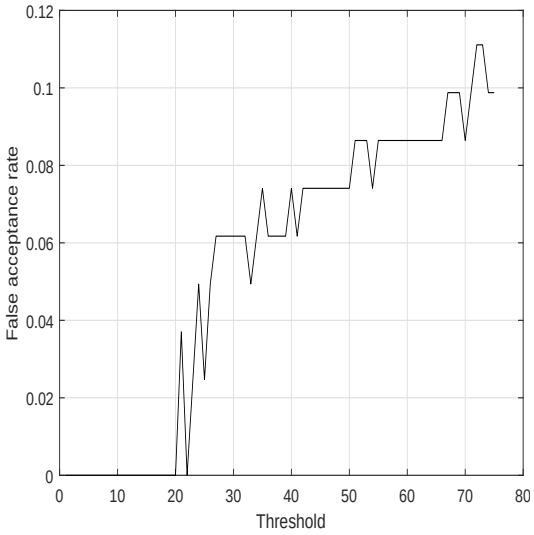


Figure 16. False acceptance rate vs. threshold.

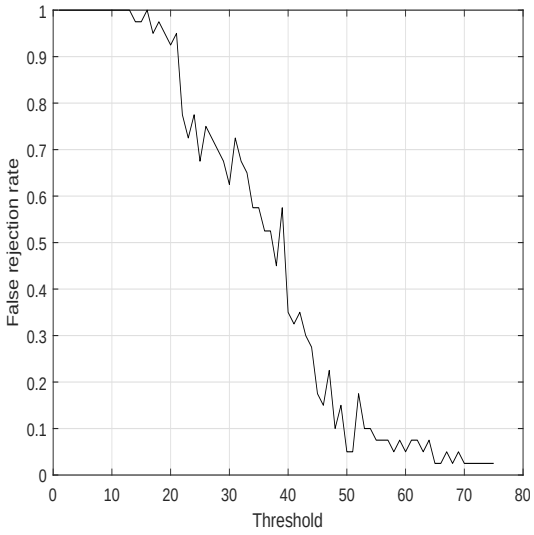


Figure 17. False rejection rate vs. threshold.

In a ROC curve, false acceptance rate and false rejection rate are plotted on the x-axis and y-axis, respectively. Both FRR and FAR are inversely proportional and there is a trade-off between these quantities. When FRR increases, then FAR decreases and vice versa. The ROC curve of the proposed scheme is plotted in Figure 18. When the FAR is very small, the value of FRR is 0.6 which is very high and unacceptable. When the threshold value is increased, the FRR start to decrease but at the same time, the value of FAR starts increasing. When the threshold value reaches 65, the FAR start increasing but at the same time, the FRR becomes constant. A value of 65 is therefore selected as a threshold for this dataset. The values of FRR and FAR at threshold 65 are listed in Table 2. The result is promising as it suggests that at a low value of FRR, the FAR of the system is also small.

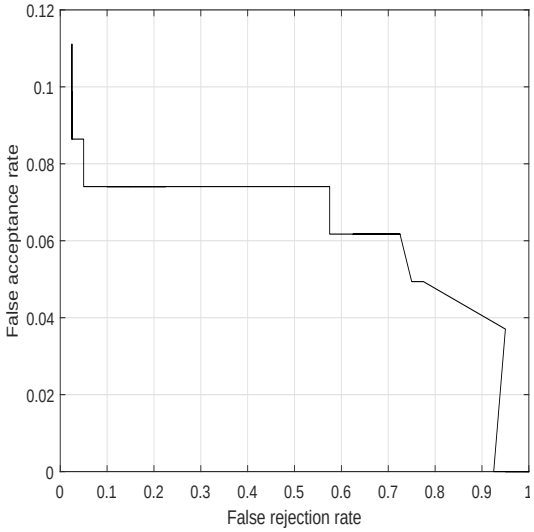


Figure 18. ROC curve showing change in *FRR* and *FAR* by varying threshold.

Table 2. Values of *FAR* and *FRR*.

Metric	Value
FAR	0.08641
FRR	0.025

6. Conclusions

In this paper, a label inspection scheme is proposed using the idea of image hashing. The proposed scheme can find misprinted areas in labels of different products. Open SURF features of both template and query images are used which makes the tamper detection robust against rotation, scaling and other geometric transformations. The MLESAC algorithm removes outliers and features which are not robust. The mean of the blocks is computed to generate the image hash. If the mean of a block is greater than the chosen threshold, then the area corresponding to that block is considered tampered. Experimental results have demonstrated that the mean value of a block which is used as the feature to generate hash is effective to find faults in printed errors in labels. Since the calculation of the mean value does not require any complex mathematical operation, therefore, the response time of the system when implemented in Python is 400 ms. Other feature extraction methods were also tested but they make the computational time slow. From the ROC curve and results shown in Table 2, it is evident that the proposed hashing scheme for label inspection is robust to geometric distortions and at the same time sensitive to detect tampering with fault localization. Although the proposed scheme is designed for label inspection, it can also be applied for authentication of natural and synthetic images.

Author Contributions: Conceptualization, F.A.; Methodology, F.A., M.A.K., J.A.; Software, M.A.K. and M.D.K.; Validation, M.A.K., M.D.K. and F.A.; Formal analysis, F.A. and M.A.K.; Investigation, M.A.K. and F.A.; Resources, F.A.; Data curation, F.A. H.K.; writing—original draft preparation, M.A.K. and F.A.; writing—review and editing, J.A., H.K. and N.P.; Supervision, F.A., J.A., N.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: One of the authors (Harish Kumar) extends his gratitude to the Deanship of Scientific Research at King Khalid University for funding this work through the research groups program under grant number R. G. P. 2/132/42.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Szeliski, R. *Computer Vision: Algorithms and Applications*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2010.
2. Swaminathan, A.; Mao, Y.; Wu, M. Robust and secure image hashing. *IEEE Trans. Inf. Forensics Secur.* **2006**, *1*, 215–230. [CrossRef]
3. Monga, V.; Evans, B.L. Perceptual image hashing via feature points: Performance evaluation and tradeoffs. *IEEE Trans. Image Process.* **2006**, *15*, 3452–3465. [CrossRef] [PubMed]
4. Monga, V.; Mihçak, M.K. Robust and Secure Image Hashing via Non-Negative Matrix Factorizations. *IEEE Trans. Inf. Forensics Secur.* **2007**, *2*, 376–390. [CrossRef]
5. Abbas, S.Q.; Ahmed, F.; Chen, Y.P.P. Perceptual image hashing using transform domain noise resistant local binary pattern. *Multimed. Tools Appl.* **2021**, *80*, 9849–9875. [CrossRef]
6. Guan, J.; Li, Y.; Sun, J.; Wang, X.; Zhao, H.; Zhang, J.; Liu, Z.; Qi, S. Graph-based supervised discrete image hashing. *J. Vis. Commun. Image Represent.* **2019**, *58*, 675–687. [CrossRef]
7. Hosny, K.M.; Khedr, Y.M.; Khedr, W.I.; Mohamed, E.R. Robust Color Image Hashing Using Quaternion Polar Complex Exponential Transform for Image Authentication. *Circuits Syst. Signal Process.* **2018**, *37*, 5441–5462. [CrossRef]
8. Du, L.; Ho, A.T.; Cong, R. Perceptual hashing for image authentication: A survey. *Signal Process. Image Commun.* **2020**, *81*, 115713. [CrossRef]
9. Gharde, N.D.; Thounaojam, D.M.; Soni, B.; Biswas, S.K. Robust perceptual image hashing using fuzzy color histogram. *Multimed. Tools Appl.* **2018**, *77*, 30815–30840 [CrossRef]
10. Tang, Z.; Zhang, H.; Lu, S.; Yao, H.; Zhang, X. Robust image hashing with compressed sensing and ordinal measures. *EURASIP J. Image Video Process.* **2020**, *2020*, 21. [CrossRef]
11. Demir, B.; Bruzzone, L. Hashing-based scalable remote sensing image search and retrieval in large archives. *IEEE Trans. Geosci. Remote. Sens.* **2015**, *54*, 892–904. [CrossRef]
12. Dai, Q.; Li, J.; Wang, J.; Chen, Y.; Jiang, Y.G. A Bayesian Hashing approach and its application to face recognition. *Neurocomputing* **2016**, *213*, 5–13. [CrossRef]
13. Ahmed, F.; Siyal, M.Y.; Abbas, V.U. A secure and robust hash-based scheme for image authentication. *Signal Process.* **2010**, *90*, 1456–1470. [CrossRef]
14. Fridrich, J.; Goljan, M. Robust hash functions for digital watermarking. In Proceedings of the International Conference on Information Technology: Coding and Computing (Cat. No. PR00540), Las Vegas, NV, USA , 27–29 March 2000; pp. 178–183.
15. Venkatesan, R.; Koon, S.M.; Jakubowski, M.H.; Moulin, P. Robust image hashing. In Proceedings of the 2000 International Conference on Image Processing (Cat. No. 00CH37101), Vancouver, BC, Canada, 10–13 September 2000; Volume 3, pp. 664–666.
16. Karsh, R.K.; Laskar, R.H. Aditi Robust image hashing through DWT-SVD and spectral residual method. *EURASIP J. Image Video Process.* **2017**, *2017*, 31. [CrossRef]
17. Karsh, R.K.; Saikia, A.; Laskar, R.H. Image authentication based on robust image hashing with geometric correction. *Multimed. Tools Appl.* **2018**, *77*, 25409–25429. [CrossRef]
18. Qin, C.; Chen, X.; Luo, X.; Zhang, X.; Sun, X. Perceptual image hashing via dual-cross pattern encoding and salient structure detection. *Inf. Sci.* **2018**, *423*, 284–302. [CrossRef]
19. Tang, Z.; Dai, Y.; Zhang, X.; Zhang, S. Perceptual image hashing with histogram of color vector angles. In Proceedings of the International Conference on Active Media Technology, Macau, China, 4–7 December 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 237–246.
20. Tang, Z.; Li, X.; Zhang, X.; Zhang, S.; Dai, Y. Image hashing with color vector angle. *Neurocomputing* **2018**, *308*, 147–158. [CrossRef]
21. Tang, Z.; Huang, Z.; Zhang, X.; Lao, H. Robust image hashing with multidimensional scaling. *Signal Process.* **2017**, *137*, 240–250. [CrossRef]
22. Ouyang, J.; Liu, Y.; Shu, H. Robust hashing for image authentication using SIFT feature and quaternion Zernike moments. *Multimed. Tools Appl.* **2017**, *76*, 2609–2626. [CrossRef]
23. Vadlamudi, L.N.; Vaddella, R.P.V.; Devara, V. Robust image hashing using SIFT feature points and DWT approximation coefficients. *ICT Express* **2018**, *4*, 154–159. [CrossRef]
24. Ng, W.W.; Li, J.; Tian, X.; Wang, H.; Kwong, S.; Wallace, J. Multi-level supervised hashing with deep features for efficient image retrieval. *Neurocomputing* **2020**, *399*, 171–182. [CrossRef]
25. Pun, C.M.; Yan, C.P.; Yuan, X.C. Robust image hashing using progressive feature selection for tampering detection. *Multimed. Tools Appl.* **2018**, *77*, 11609–11633. [CrossRef]
26. Fei, M.; Ju, Z.; Zhen, X.; Li, J. Real-time visual tracking based on improved perceptual hashing. *Multimed. Tools Appl.* **2017**, *76*, 4617–4634. [CrossRef]
27. Ji, J.; Yao, Y.; Wei, J.; Quan, Y. Perceptual hashing for SAR image segmentation. *Int. J. Remote Sens.* **2019**, *40*, 3672–3688. [CrossRef]

28. Yang, B.; Gu, F.; Niu, X. Block mean value based image perceptual hashing. In Proceedings of the 2006 International Conference on Intelligent Information Hiding and Multimedia, Pasadena, CA, USA, 18–20 December 2006; pp. 167–172.
29. Babenko, B.; Yang, M.H.; Belongie, S. Robust object tracking with online multiple instance learning. *IEEE Trans. Pattern Anal. Mach. Intell.* **2011**, *33*, 1619–1632. [CrossRef] [PubMed]
30. Ma, J.; Ma, Y.; Li, C. Infrared and visible image fusion methods and applications: A survey. *Inf. Fusion* **2019**, *45*, 153–178. [CrossRef]
31. Ma, J.; Zhao, J.; Tian, J.; Tu, Z.; Yuille, A.L. Robust estimation of nonrigid transformation for point set registration. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 23–28 June 2013; pp. 2147–2154.
32. Wang, Z.; Yi, P.; Jiang, K.; Jiang, J.; Han, Z.; Lu, T.; Ma, J. Multi-memory convolutional neural network for video super-resolution. *IEEE Trans. Image Process.* **2019**, *28*, 2530–2544. [CrossRef]
33. Ryu, S. Local Area Transform for Cross-Modality Correspondence Matching and Deep Scene Recognition. *arXiv* **2019**, arXiv:1901.00927.
34. Swain, M.J.; Ballard, D.H. Color indexing. *Int. J. Comput. Vis.* **1991**, *7*, 11–32. [CrossRef]
35. Schiele, B.; Crowley, J.L. Object recognition using multidimensional receptive field histograms. In Proceedings of the European Conference on Computer Vision, Cambridge, UK, 14–18 April 1996; Springer: Berlin/Heidelberg, Germany, 1996; pp. 610–619.
36. Murase, H.; Nayar, S.K. Visual learning and recognition of 3-D objects from appearance. *Int. J. Comput. Vis.* **1995**, *14*, 5–24. [CrossRef]
37. Lowe, D.G. Object recognition from local scale-invariant features. In Proceedings of the Seventh IEEE International Conference on Computer Vision, Kerkyra, Greece, 20–27 September 1999; Volume 2, pp. 1150–1157.
38. Bay, H.; Tuytelaars, T.; Van Gool, L. Surf: Speeded up robust features. In Proceedings of the European Conference on Computer Vision, Graz, Austria, 7–13 May 2006; Springer: Berlin/Heidelberg, Germany, 2006; pp. 404–417.
39. Torr, P.H.; Zisserman, A. MLESAC: A new robust estimator with application to estimating image geometry. *Comput. Vis. Image Underst.* **2000**, *78*, 138–156. [CrossRef]
40. Alcantarilla, P.F.; Bartoli, A.; Davison, A.J. KAZE features. In Proceedings of the European Conference on Computer Vision, Florence, Italy, 7–13 October 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 214–227.
41. Matas, J.; Chum, O.; Urban, M.; Pajdla, T. Robust wide-baseline stereo from maximally stable extremal regions. *Image Vis. Comput.* **2004**, *22*, 761–767. [CrossRef]
42. Leutenegger, S.; Chli, M.; Siegwart, R. BRISK: Binary robust invariant scalable keypoints. In Proceedings of the 2011 IEEE International Conference on Computer Vision (ICCV), Barcelona, Spain, 6–13 November 2011; pp. 2548–2555.