applied sciences

MDPI

Article

Texture Construction Edge Detection Algorithm

Shou-Cih Chen and Chung-Cheng Chiu *

Department of Electrical and Electronics Engineering, Chung Cheng Institute of Technology, National Defense University, Taoyuan 33551, Taiwan; e6986715@gmail.com
* Correspondence: david.cc.chiu@gmail.com; Tel.: +886-3-390-9962

Received: 29 January 2019; Accepted: 26 February 2019; Published: 3 March 2019

Abstract: The edge detection algorithm is the cornerstone of image processing; a good edge detection result can further extract the required information through rich texture information and achieve object detection, segmentation, and identification. To obtain a rich texture edge detection technology, this paper proposes using edge texture change for edge construction and constructs the edge contour through constructing an edge texture extension between the blocks to reduce the missing edge problem caused by the threshold setting. Finally, through verification of the experimental results, the proposed method can effectively overcome the problem caused by unsuitable threshold setting and detect rich object edge information compared to the adaptive edge detection method.

Keywords: edge detection; edge thinning; texture blocks; texture template

1. Introduction

Object contours play an important role in human visual processing and their composition presents some meaningful geometric concepts. By defining the contour shape, people can effectively and widely identify various objects. The contour of an object does not change with color or ambient brightness, so the object information is stable. It is widely used in object segmentation, object detection, texture identification, and feature analysis, and is popular in fields such as security monitoring systems, smart home systems, and smart vehicles. Besides the shape of the outer contour of the object, subtle texture changes inside the object also imply important object information, so dominant or implicit texture changes should be selected or retained as each possible object identification feature.

Edge detection technology is the main method used to detect the contour of objects. Recently, many experts and scholars have been working on various image edge detection technologies. However, finding rich object edges from target images is still a challenging and popular topic. The traditional edge detection technology developed in the early days mainly aimed to find the discontinuity in the gray-level intensity of a pixel. After obtaining the gradient information by a first- or second-order differential operation, the intensity difference between the center and adjacent pixels was observed and analyzed to obtain the edge. For example, Robert's operator [1] uses a 2×2 mask to calculate the difference between adjacent pixels in the diagonal direction to obtain the gradient information and determines the edge retention through a threshold setting. Because the mask used is 2×2 in size and has no clear center, resulting in an inaccurate range of values, the Prewitt operator [2] was developed, which uses a 3×3 mask for gradient operations, combining horizontal and vertical gradients. In this operator, the component obtains the gradient information of the entire image, and finally, retains the edges through a threshold setting. The Sobel [3] operator is similar to the Prewitt [2] operator. One slight difference between the operators is that the mask center uses the weighting coefficient value and some noise suppression is achieved by giving the center point a larger weighting value. Finally, one still has to keep the edges through a threshold setting. Because the setting of the threshold value directly affects the output of the edge detection, some scholars have proposed an adaptive threshold value

method to solve the error caused by manual adjustment. Bora and Gupta [4] proposed the combination of Otsu [5] and Sobel [3] algorithms to achieve image filtering, and used the Watershed algorithm to achieve image segmentation. Davis et al. [6] used the adaptive threshold method proposed by Bradely and Roth [7] to calculate the percentage of the average grayscale value of the mask using the integral image, as compared to the center pixel, as the setting of the image binarization threshold, and finally, applied the binarized image to object identification. Luo et al. [8] calculated and retained the maximum of the two statistical values as the basis for determining the edge retention of the adaptive threshold. One is the sum of the gradient average and the one-half standard deviation in the mask and the other is a custom threshold. The calculation of the adaptive threshold value significantly improves the detection effect of the edge detection algorithm, and effectively avoids the error caused by manual operation. However, since the acquisition of a single threshold value only depends on the statistical result, when the image grayscale variations are not obvious, the high-threshold setting may cause the less-obvious edge texture to be filtered out. On the contrary, the low-threshold setting may cause too much noise or too many false edges to be retained.

To effectively reduce the problem of edge discontinuity or loss, the Canny algorithm [9] proposes using high-threshold and low-threshold values instead of a single threshold. Since the higher gradient value is a more pronounced edge, Canny uses the hysteresis effect to track pixels that are held between the high and low thresholds, as possible for object edge retention. To find the best threshold, some scholars have proposed a method for detecting high and low thresholds for Canny [9] edge detection; for example, Gao and Liu [10] used the Otsu method for the gradient magnitude histogram after non-maximum suppression. The calculated gradient magnitude histogram is used to obtain the high threshold and sets one-half of the high threshold as a low threshold. However, since the low threshold setting is determined by the high threshold, it has not been accurately calculated by statistical analysis. Therefore, it is inaccurate in terms of the setting, which may still lead to edge loss. Song et al. [11] also used the Otsu method to analyze the gradient magnitude histogram after non-maximum suppression. They first obtained the high-threshold value and then excluded the gradient value greater than the high threshold and performed the second operation on the statistical result to obtain a low threshold. The proposed method of the iteration threshold effectively improves the edge detection discontinuity and detection errors. Saheba et al. [12] proposed using the mean squared error (MSE) [13] to automatically detect the high and low thresholds. First, one uses the MSE to quantify the gradient magnitude histogram after non-maximum suppression. To perform the operation, one first obtains the high threshold, then excludes the gradient larger than the high threshold, and finally performs the second operation on the statistical result to obtain the low threshold. The proposed method improves the edge detection effect in high- and low-brightness images. In addition, the gradient magnitude histogram proposed by Li and Zhang [14] uses a differential operation to obtain the adaptive high and low thresholds after non-maximum suppression. The high threshold is obtained by the first zero bin of the amplitudes difference, and the low threshold is set to 0.4 times the high threshold. The method improved the ability of noise reduction and made the edge more obvious, but the low threshold is still dependent on the calculation of the high-threshold value and the setting is less accurate and may still lead to loss of the edge. Ferdous et al. [15] used the method proposed by Rupalatha et al. [16] to adaptively solve the high and low thresholds by the operation of two statistical values in the gradient histogram after non-maximum suppression: one is the average value of the probability density function and the other is the variation in the probability density function. The proposed method has better continuity at the detected edge. Although the Canny [9] algorithm uses a high- and low-threshold strategy to preserve the edges more flexibly, there is still a common problem of threshold setting. A pixel with a gradient value larger than the high-threshold is used as the starting point of the edge. Therefore, the high-threshold value directly affects the edge detection result and the low-threshold value is the lower limit of the gradient value of the edge detection. When the threshold value is set too high, the edge texture with a lower gradient value is filtered out. Conversely, when the setting is too low, too much noise or too many false edges are left. Because of the variability of

the image, a fixed set of high and low thresholds cannot effectively be used to preserve the rich edge information in the image. Therefore, finding the edge of the object in the image by setting the threshold value is an insurmountable problem.

To solve the excessive or effective edge texture loss caused by unsuitable threshold value setting, the texture construction edge detection algorithm (TCEDA) is proposed as the edge construction algorithm. First, an image preprocessing is used to reduce the noise interference caused by the external environment effects, such as illumination, shooting angle, and visual attributes. Then, using non-maximum suppression and the thinning texture template, the image output of the edge thinning process is achieved, and finally, the texture of the object's edge is extended by the block scanning mask to realize the construction of the edge texture of the object. The proposed TCEDA can obtain richer edge detection results as the basis for subsequent object detection, segmentation, and identification. The rest of the paper is organized as follows: Section 2 introduces the TCEDA and process architecture; Section 3 shows the experimental results of the TCEDA and the adaptive threshold value algorithms; and Section 4 summarizes the contribution and follow-up of the TCEDA and applications.

2. Texture Construction Edge Detection Algorithm

This section explains how the TCEDA proposed in this paper uses the block texture change to extend the edge information of the image. When there is a grayscale change between the adjacent pixels, it will be included in the algorithm to determine whether to retain it as the edge of the image of the texture information. As to how to build the edge through the effective block texture feature, this paper distinguishes it in the following three parts: image pre-processing, the optimal edge thinning process, and edge texture construction processing. First, the image pre-processing converts the input color image into a grayscale image. In addition, to reduce the noise interference caused by the imaging technology and the operation of the optical device during the acquisition process, a Gaussian smoothing filter is used to weaken the variation of the grayscale value between the adjacent pixels in the image and the Sobel operator is used to calculate the gradient values of the image. Then, in order to filter the invalid texture variation features, the redundant pixels and weaker edges of the gradient change are regarded as noise. Therefore, a fixed threshold is set in the algorithm to filter noise to reduce its interference with the contour edges of the details in the image. Second, for the optimal edge thinning process, to avoid excessive unnecessary analysis of the image pre-processing results and reduce the complexity of the subsequent object edge construction, the algorithm uses the non-maximum suppression method to find pixels with larger regional gradient values. The point is used as a candidate edge point and the thinning texture template is then used to filter out redundant pixels to achieve the optimal thinning effect. Finally, in the edge texture construction processing part, since the texture change with a long extension has more meaningful edge features, the algorithm establishes the edge texture through the connection between the central block and the four adjacent blocks. The steps in the algorithm are described in detail in the subsections below.

2.1. Image Preprocessing

Since the grayscale image retains most of the edge information in the image, the color image containing the three color components R (red), G (green), and B (blue) is first converted into the grayscale image. The grayscale image is gray-scaled by the commonly used grayscale formula, which is expressed as follows:

$$\text{Gray} = 0.299 \times \text{R} + 0.587 \times \text{G} + 0.114 \times \text{B} \quad (1)$$

Image capture will be affected by imaging application-related technologies, such as the generation of scattered noise due to a dark current during the operation of the optical device, error propagation caused by the chromaticity coordinate reading and the color space conversion process, shooting angle, and external environmental factors such as the rough surface of the object affected by light and shadow, so the generation of noise is inevitable in the process of image acquisition. Therefore, to reduce

the noise caused by the above reasons, the image edge is obviously disturbed. First, the grayscale image, $f(x, y)$, is smoothed using a two-dimensional Gaussian function filter. The Gaussian function aims to calculate the transformation of each pixel in the image by using a normal distribution. The two-dimensional normal distribution equation $h(x, y)$ is defined as follows:

$$h(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2}$$

where σ (preset value is 1) is the standard deviation of the normal distribution. After image smoothing is completed, the filter mask of Sobel [3] is used to find the gradient value of each pixel in the image, using a 3×3 horizontal mask h_x, as Equation (3), and the vertical mask h_y, as Equation (4). After obtaining the gradient values in the horizontal and vertical directions (i.e., G_x and G_y), the gradient amplitude and angle of each pixel are obtained by Equations (5) and (6) to obtain the results of the gradient value (grad f) and the gradient angle (θ), respectively. The relevant operation formula is expressed as follows:

$$h_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{3}$$

$$h_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{4}$$

$$\text{grad } f = \text{mag}(\nabla \text{G}) = \left[G_x^2 + G_y^2\right]^{\frac{1}{2}} \tag{5}$$

$$\theta = \tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{6}$$

Through the gradient amplitude obtained by the above operation, the change between the adjacent pixels in the image can be observed. This algorithm is included as an effective block texture feature when the pixel gradient amplitude is not zero. Therefore, when the pixel displays any gradient amplitude change in the process, it will be included in the subsequent algorithms to be dealt with. Therefore, to avoid excessive interference of the edge texture in the image to establish an effective block edge texture feature, the pixel with a smaller gradient amplitude is regarded as noise in the algorithm and a fixed threshold value is set to filter. In addition, the analysis is retained for further analysis by retaining valid texture features. After several image experiments, the gradient threshold parameter is set to 15. Figure 1 shows the process through which the algorithm uses a fixed threshold to filter out noise. Figure 1a shows the original image, and there are many inconspicuous faint light and shadow changes in the roof, door panel, and wall surface. Figure 1b shows the binarized image from Figure 1a, with a gradient amplitude greater than zero after image pre-processing, which produces considerable noise texture due to noise generation and light and shadow changes. Figure 1c shows the image results obtained from Figure 1b after filtering the preset threshold value, which effectively filters out the light and shadow changes that are difficult to observe by most naked eyes and the noise generated by imaging technology limitations and does not have a significant impact on the main edge texture changes. After filtering through the preset threshold, most of the non-obvious noise in the image is still preserved. Further algorithmic methods can be used to filter out more noise and preserve effective edge texture changes by the following processing.

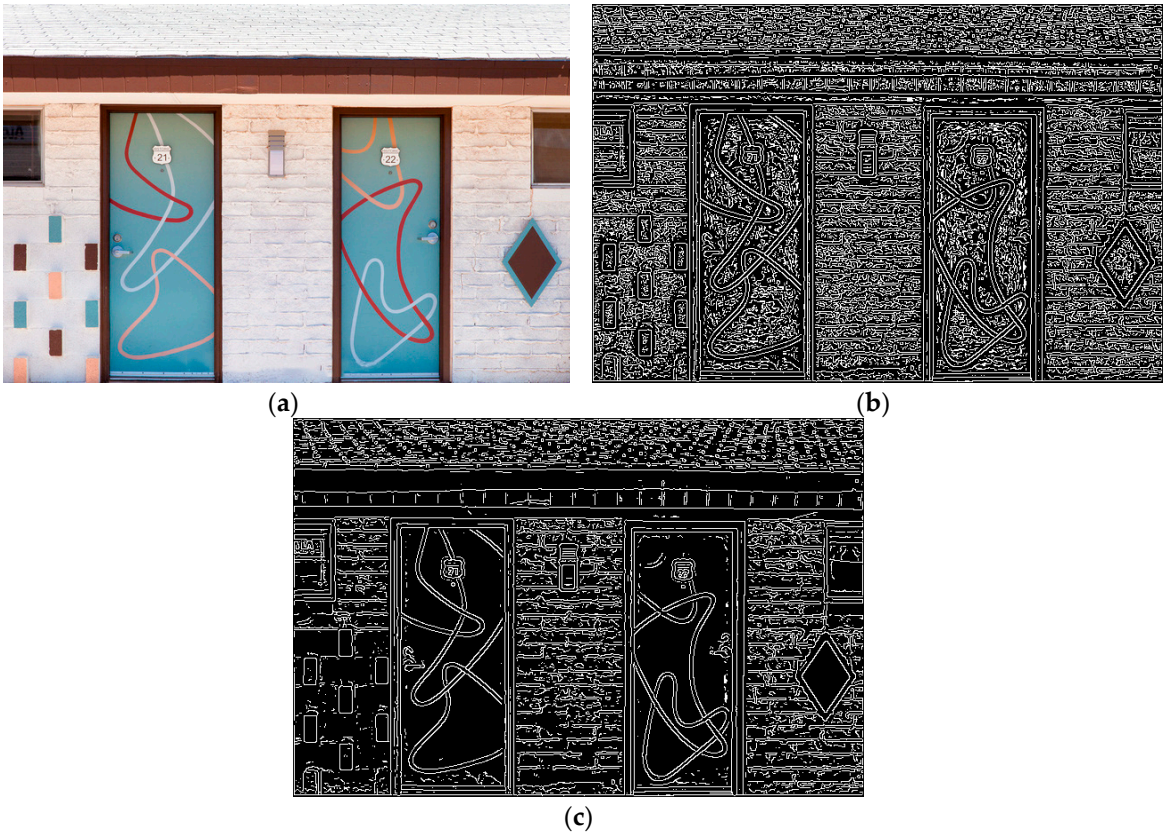


Figure 1. Noise filtering process. (**a**) Original image (640 × 427) [17]; (**b**) gradient change image before noise filtering; (**c**) gradient change image after noise filtering.

2.2. Optimal Edge Thinning Process

The optimal edge thinning process includes the two steps of non-maximum suppression and the thinning texture template process to achieve optimal edge thinning. Since the edge segments obtained by the pre-processing steps have different thicknesses, to simplify the subsequent analysis and determine the texture change combination required for edge construction, the edge thinning processing is indispensable.

First, the non-maximum suppression method uses the gradient angle obtained by Equation (6) and divides it into four directions, according to the degree of approximation: 0° (i.e., horizontal), 45° (i.e., upper right and lower left), 90° (i.e., vertical), and 135° (i.e., upper left and lower right). Then, it compares the gradient amplitude of the two pixels adjacent to the center pixel according to the gradient direction of the center pixel. When the gradient amplitudes of the two adjacent points are smaller than that of the center pixel, the center pixel is included as a candidate edge point. Conversely, the center pixel is not included as a candidate edge point. Although the method of non-maximum suppression can achieve preliminary thinning processing, there are still many redundant pixels in the candidate edge image after non-maximum suppression.

To filter the redundant pixels processed by the maximum suppression method, a block texture distribution with a center pixel as a redundant pixel is proposed to develop a set of thinning texture templates and a thinning texture template is used to filter out redundant pixels to achieve optimal edge thinning. The redundant pixel filtering process uses a raster-scan method to perform a 3 × 3 pixels block texture comparison on the non-maximum suppressed candidate edge image. When the block texture matches the defined thinning texture template, the redundant pixel in the center of the block is deleted. Figure 2 shows an example of the thinning process.

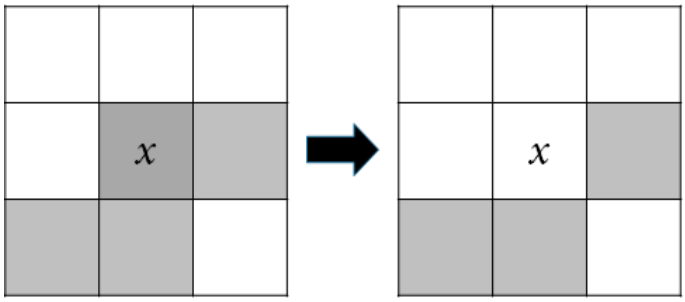


Figure 2. Thinning process example.

When the block texture of the 3 × 3 pixels covered by the center pixel x conforms to the proposed thinning texture template, it is judged as a redundant pixel and gets filtered out. Then, it gets right-shifted to make the judgment of the block texture change of the next point, until the processing of the entire candidate edge image is completed; thus, the image with optimal edge thinning can be obtained.

Next, a method of binarizing the edge texture changes in the block will be described to facilitate subsequent analysis and processing. Figure 3 illustrates the binary coding method showing the change in block texture. When the pixel gradient amplitude value is not 0, the code is "1", and otherwise, it is "0". The coding order starts from the upper left corner of the block and goes in order from left to right and top to bottom.

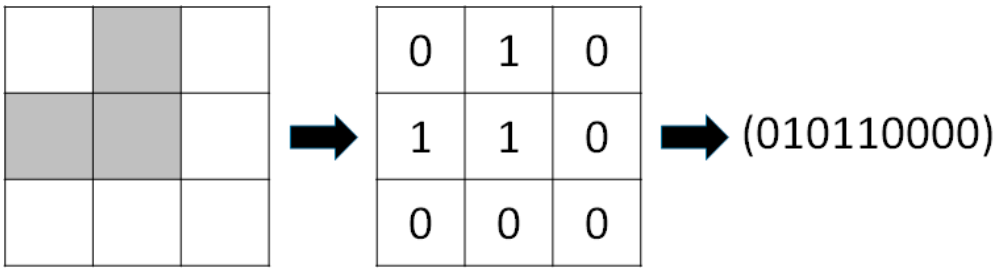


Figure 3. Binary coding.

There are up to four texture changes through angle rotation, i.e., 0°, 90°, 180°, and 270°, in the block. Therefore, after the process of binarization coding is completed, the block binarization coding of the four rotation angles is obtained by changing the bit arrangement order. Figure 4 shows the coding order of the different angles in the block. Figure 4a shows the block coding order of 0° and a coding order of 123456789. Figure 4b shows the block coding order of Figure 4a with a rotation of 90° to the left and a coding order of 369258147. Figure 4c shows the block coding order of Figure 4a with a rotation of 180° to the left and a coding order of 987654321. Figure 4d shows the block coding of Figure 4a with a rotation of 270° to the left and a coding order of 741852963.

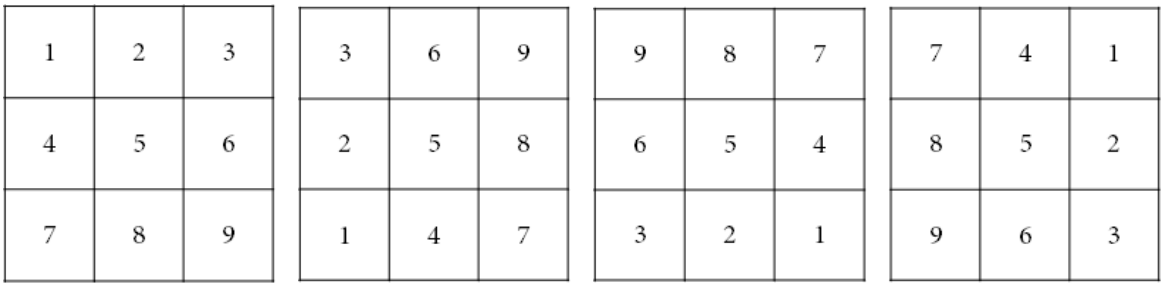


Figure 4. Binary coding order of different angles in a block. (**a**) 0°; (**b**) 90°; (**c**) 180°; (**d**) 270°.

The binarized block texture template of different angles can be obtained using the binarized block texture template of 0°. Figure 5 shows the schematic of a block texture template that uses a permutation order of the binarized coding order to obtain different angles. Figure 5a shows a texture change at 0° and a binarization code of 000011110. Figure 5b shows a texture change with Figure 5a rotated by 180° to the left and a binarization code of 011110000. Therefore, in the subsequent thinning texture template, only the 0° binarized block texture must be retained and the remaining three angle changed binarized block textures can be obtained by different coding orders.

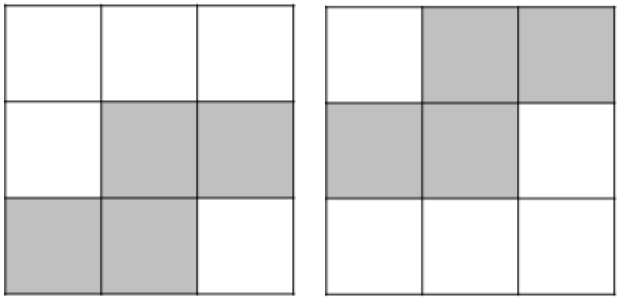


Figure 5. Block texture template angle change. (**a**) 0°; (**b**) 180°.

The binarized coding of the thinning texture template developed at the table is shown in Table 1. According to the number of pixels and the distribution, it can be summarized into 31 patterns and sorted according to the size of the binarized value. Among them, there are 28 types of templates with four angle changes, one type of template with two angle changes, and two types of templates with a single angle change. The four angle changes refer to the four different textures of the template containing 0°, 90°, 180°, and 270° of angle rotation; the two angle changes refer to the two different textures of the pixels in the template containing 0° and 90° angle rotation; and the single angle change means that the template contains a texture of 0°.

Table 1. Thinning texture template binarization coding.

Number of Pixels	4 Changes in Degree (0°, 90°, 180°, and 270°)			2 Changes in Degree (0° and 90°)	1 Change in Degree (0°)
3	000110010				
4	000010111 000110110	000011110 000111010	000110011		
5	000011111 000111110 010111001	000110111 001011110	000111011 001111010		010111010
6	000111111 001111110 011111001	001011111 010111011 110111001	001111011 010111101		
7	001111111 111110011	010111111 111111001	011110111	011111110	
8	011111111	101111111			
9					111111111

Next, a processing method for filtering out redundant pixels of the candidate edge images processed by the non-maximum suppression method according to the thinning texture template prepared will be described. First, the edge point of the candidate edge image is taken as the center pixel and a block of 3 × 3 pixels is taken out. When there is one pixel in the block, it is regarded as isolated noise filtering; when the block has two pixels, it is regarded as the end of the edge segment; and when there are three or more pixels in the block, the thinning texture template in Table 1 and its angled rotation are compared to see whether the texture changes match according to the number of pixels in the block. When they match, the redundant pixels in the candidate edge image are filtered out sequentially in the raster-scan process to achieve the best edge thinning process.

Figure 6 shows the preliminary thinning candidate edge image obtained in Figure 1c after the non-maximum suppression method. From the partial enlargement area, it can be observed that there are still redundant pixels around most of the edges, so it does not achieve the optimal edge thinning treatment. Figure 7 shows the best edge thinning image obtained after the thinning texture template processing. Through the partial enlargement area, it can be clearly seen that the image edge is composed of only a single pixel, which can effectively filter pixels that have not performed the maximum suppression. There are still redundant pixels around the image edges to achieve optimal edge thinning. The total number of candidate edge points in Figure 6 is 45,608, and the total number of optimal thinning edge points in Figure 7 is 38,835. In Figure 7, when compared to Figure 6, the number of candidate edge points is reduced by 6773, which is about 15% lower than the number of edge points. The optimal edge thinning results obtained after processing will facilitate the analysis and processing of the subsequent edge texture construction.

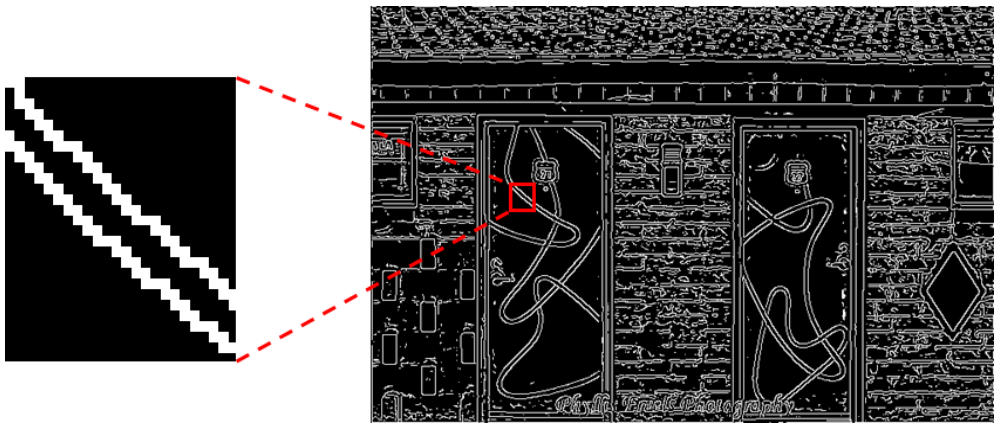


Figure 6. Candidate edge image after non-maximum suppression.

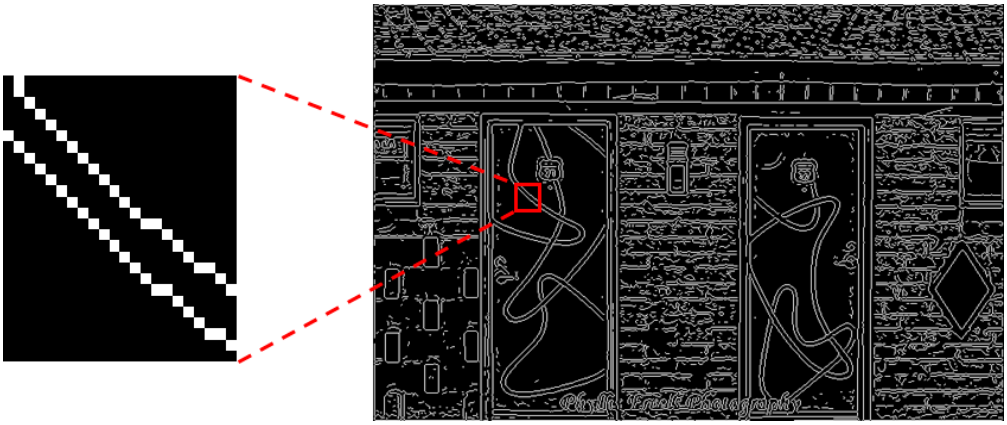


Figure 7. Edge image with optimal edge thinning.

2.3. Edge Texture Construction Processing

After the optimal edge thinning process step is completed, the edge image of the component is extended by the block edge texture template processing for the edge image of the optimal edge thinning. Since the extended texture change has more meaningful edge features, the edge texture construction processing is extended by expanding the edge texture template created between adjacent blocks, which defines a single edge segment as having at least six pixels. With the above extension length, the edge texture construction processing marks the edge points that satisfy the abovementioned conditional edge texture changes as valid points. Figure 8 shows the block texture scan mask of the present process, which is composed of five adjacent 3 × 3 pixels blocks, based on the relation

between the edge points of number *x1–x5* in the central block *X*, where the center pixel *x* is located, and the adjacent blocks *P*, *Q*, *R*, and *S*, as well as the change in the texture length, to preserve the edge texture information.

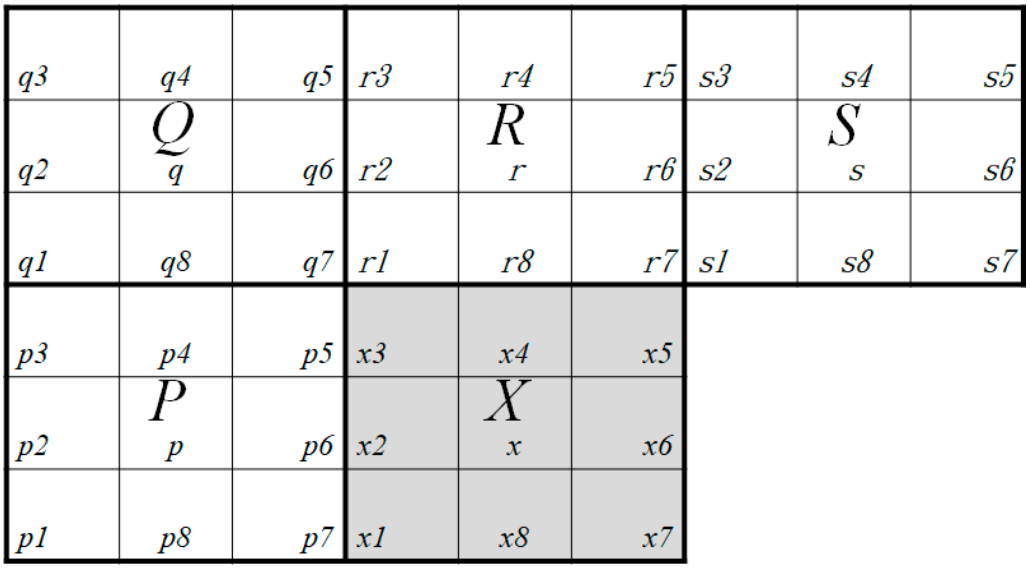


Figure 8. Block texture scan mask.

The edge texture construction processing aims to use the block texture scan mask as the range to process the edge image with the optimal edge thinning process of the raster-scan method. In order to preserve the effective edge texture information in the image after the optimal edge thinning process, the process scans all edge points pixel-by-pixel and sequentially marks the edge points that meet the conditions. All edge points marked after scanning the entire image are the final edge detection results. Figure 9 shows the flow of the proposed edge texture construction processing.

Each process of the edge texture construction processing is sequentially described in the flowchart of Figure 9. First, the image with optimal edge thinning is used as the source image input, and then it is determined whether all edge points in the source image are processed completely. When all edge points have been processed, the process ends and the edge image with the edge texture mark is obtained. Conversely, when there is an unprocessed edge point, the raster-scan sequence is moved to the next edge point and the subsequent pixel is processed with the point as the center pixel.

Then, it is judged whether the nine points in the *X* block, where the center pixel is located, have edge points marked as valid points. When there are edge points marked as valid points, the process proceeds to the next process to perform block *X* edge point processing. The edge point of the valid point in *x1–x5* is first deducted, and one records the position of all edge points in the *X* block and proceeds to the next process to determine whether there is an edge point in *x1–x5*. When there is no edge point, one goes to the marking processing step to process all edge points in the *X* block; when there are still edge points, the block texture extension analysis is performed in the next step of the flowchart. When no edge points of the nine points in the *X* block are marked as valid points, the number of edge points of the *X* block in which the center pixel is located is calculated: when the block contains at least three (inclusive) or more edge points, it goes to the block texture extension analysis; when there are less than three edge points in the block, the endpoint of the edge segment is regarded as unmarked, the initial step of the edge texture construction processing is returned to, and the next edge point is processed.

Next, the method of block texture extension analysis is described as a whole. The block texture extension analysis is based on the block scan mask range of Figure 8. Through the *X* block where the center pixel is located and its extended connection among the four adjacent blocks *P*, *Q*, *R*, and *S*, horizontal, vertical, and diagonal edge texture changes are formed. During the processing, the texture

of the adjacent block is first determined. The position of the edge point of *x1–x5* in the *X* block is used to determine whether there are adjacent points in the adjacent block and the following three cases are distinguished.

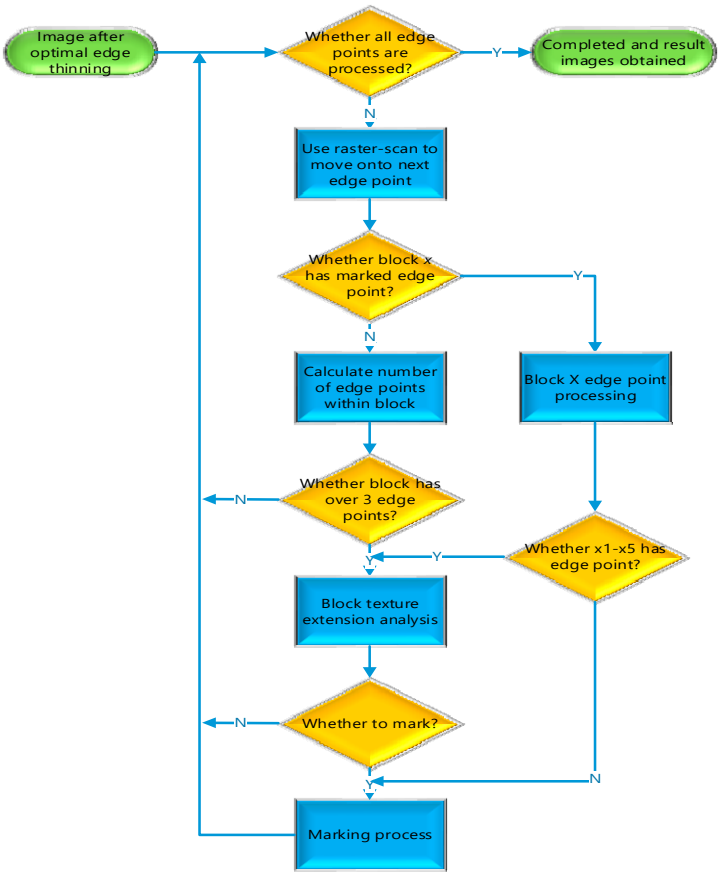


Figure 9. Edge texture construction process flowchart.

In the first case, when there is no adjacent point in the adjacent block, the marking process is not performed and the block texture extension analysis is completed, and the next step of the flowchart is performed. In the second case, when adjacent blocks have adjacent points and any adjacent point is marked as valid, the *x1–x5* edge points adjacent to the valid points must be deducted; meanwhile, the positions of all edge points in the *X* blocks are additionally recorded and it is judged whether there are edge points in *x1–x5*. When there is no edge point, the block texture extension analysis is completed, all edge points of the *X* block to be marked are recorded, and the next step of the flowchart is followed. When there are still edge points left, one checks whether there are adjacent points in the adjacent blocks: when there is no adjacent point, the first case is processed; when there are adjacent points, the third case is processed to perform the analysis.

In the third case, when the adjacent blocks have adjacent points, but the adjacent points are not marked as valid, the table lookup and comparison are performed according to the texture distribution of the adjacent blocks and all block texture templates that may meet the length of the effective edge line segment, as shown in Tables 2–15. To speed up the table lookup process, the block texture template converts the binarized code in the block into a decimal number, which is sorted from small to large. The comparison process is based on the binary search method [18]. If the comparison result is consistent, it means that the central and adjacent blocks have an edge texture change that satisfies the condition. Then, the block texture extension analysis is completed and the positions of all edge points in the *X* block and the adjacent block to be marked are recorded. It then proceeds to the next step of the flowchart. On the other hand, if the comparison result does not match, the cross-block texture extension

judgment is performed, where the position of the edge point of *x1–x5* in the *X* block is used as the starting point and one searches for edge points in the area adjacent to 8 that does not contain the *X* block. When there are less than three points in the adjacent area and none of them are marked as valid, the *X* block and the adjacent area are not marked. The block texture extension analysis is then completed and the next step of the flowchart is followed; when the adjacent area is extended to an area where three edge points (inclusive) are marked as valid, the block texture extension analysis is completed and all required positions in the *X* block and adjacent areas are recorded, and the next step of the flowchart is followed; when the number of adjacent edge points reaches three, the block texture extension analysis is completed and all required positions in the *X* block and adjacent areas are recorded, and the next step of the flowchart is followed.

Table 2. *x1* corresponding to block *P* texture template (decimal).

7	14	19	21	28	35	39	42	46	49	53	56	60	71	73
74	77	78	81	83	85	88	92	99	105	106	109	113	115	117
120	135	137	138	141	142	145	147	149	156	163	167	168	169	170
172	173	177	199	200	201	202	204	205	206	209	213	227	231	232
233	236	237	241	263	270	273	275	277	280	282	284	291	295	298
302	305	312	327	329	330	333	334	337	339	341	344	355	361	362
365	369	391	392	393	394	396	397	398	401	405	419	424	428	455
456	457	458												

Table 3. *x2* corresponding to block *P* texture template (decimal).

7	14	19	21	28	35	39	42	46	49	53	56	60	71	74
78	81	82	83	84	85	86	99	106	112	113	114	115	116	117
135	137	138	139	141	142	143	145	147	149	156	163	167	168	169
170	172	173	177	199	202	206	208	209	210	212	213	224	225	226
227	228	229	231	241	263	270	273	275	277	280	282	284	291	295
298	302	305	312	327	330	334	336	337	338	339	340	341	355	362
368	369	391	392	393	394	396	397	398	401	405	419	424	428	448
449	450	451	452	453	454	455	458	480	481	482	484			

Table 4. *x3* corresponding to block *P* texture template (decimal).

11	14	15	25	28	29	42	43	46	56	57	60	73	74	75
77	78	79	81	82	83	84	85	86	105	106	107	109	110	112
113	114	116	117	137	138	139	141	142	143	156	168	169	170	172
173	208	209	210	212	213	224	225	226	227	228	229	267	270	280
281	282	284	298	302	312	329	330	333	334	336	337	338	339	340
341	342	344	361	362	363	365	368	369	370	392	393	394	396	397
398	424	425	428	448	449	450	451	452	453	454	455	480	481	482
483	484													

Table 5. *x3* corresponding to block *Q* texture template (decimal).

7	11	15	19	21	25	29	35	39	43	49	53	57	71	73
75	77	79	81	83	85	99	105	107	109	113	115	117	135	137
139	141	143	145	147	149	163	167	169	173	177	199	201	205	209
213	227	231	233	237	241	263	273	275	277	281	285	291	295	302
305	327	329	333	337	339	341	355	361	363	365	369	391	393	397
401	405	419	455	457										

Table 6. *x3* corresponding to block *R* texture template (decimal).

7	11	14	15	19	21	28	29	35	42	43	52	53	60	71
74	75	78	79	82	83	84	85	92	99	106	107	114	116	117
135	138	139	142	143	146	147	148	149	156	157	162	163	164	165
170	172	173	181	199	202	210	212	213	226	227	228	229	236	263
267	270	271	274	275	276	277	282	284	285	290	291	292	293	298
299	300	301	327	330	334	338	340	341	348	354	355	356	357	362
364	365	391	394	395	398	399	402	404	405	418	420	428	429	455
458	482	484												

Table 7. *x4* corresponding to block *R* texture template (decimal).

14	21	25	28	29	35	42	43	46	49	52	53	57	60	73
74	77	78	81	82	84	85	92	99	105	106	107	109	113	114
116	117	137	138	141	142	145	146	148	149	156	157	162	163	164
165	169	170	172	173	177	181	201	202	205	209	210	212	213	226
227	228	229	233	236	270	273	274	276	277	281	282	284	285	290
291	292	293	298	299	300	301	302	305	329	330	333	334	337	338
340	341	348	354	355	356	357	361	362	364	365	393	394	397	398
401	402	404	405	418	420	428	429	457	458	482	484			

Table 8. *x5* corresponding to block *R* texture template (decimal).

7	14	21	22	25	29	35	38	39	42	46	49	53	57	71
73	74	77	78	81	82	83	85	86	99	102	105	106	107	109
113	114	117	135	137	138	141	142	145	146	149	150	162	163	166
167	169	173	177	199	201	202	205	206	209	210	226	227	231	231
233	237	241	263	270	273	274	277	278	281	282	285	290	291	294
295	298	302	305	327	329	330	333	334	337	338	341	354	355	361
362	365	369	391	393	394	397	398	401	402	405	418	419	455	457
458	482													

Table 9. *x5* corresponding to block *S* texture template (decimal).

7	14	15	21	22	28	29	38	39	46	52	53	60	71	78
79	84	85	86	92	116	117	135	142	143	148	149	150	156	157
164	165	166	167	172	173	181	199	212	213	228	229	231	236	263
270	271	276	277	278	284	285	292	293	294	295	300	301	302	327
334	340	341	342	348	356	357	364	365	391	398	399	404	405	420
428	429	455	482	484										

Table 10. *x1* and *x2* corresponding to block *P* texture template (decimal).

7	19	71	81	82	83	84	85	86	99	112	113	114	115	116
117	167	199	208	209	210	212	213	224	225	226	227	228	229	231
241	327	336	337	338	339	340	341	355	368	369	448	449	450	451
452	453	454	455	480	481	482	484							

Table 11. *x1* and *x3* corresponding to block *P* texture template (decimal).

7	14	19	21	25	28	29	35	39	42	46	49	53	56	57
60	71	73	74	77	78	81	82	83	84	85	86	88	92	99
105	106	109	110	112	113	114	116	117	120	135	137	138	141	142
145	147	149	156	163	167	168	169	170	172	173	177	199	208	209
210	212	213	224	225	226	227	228	229	233	263	270	273	275	277
280	281	282	284	291	295	298	302	305	312	327	329	330	333	334
336	337	338	339	340	341	342	344	346	348	355	361	362	365	368
369	370	391	392	393	394	396	397	398	401	419	424	425	428	448
449	450	451	452	453	454	455	457	480	481	482	483	484		

Table 12. *x2* and *x3* corresponding to block *P* texture template (decimal).

7	19	21	35	39	49	53	71	81	83	85	86	99	113	115
117	135	145	147	149	163	167	177	199	209	213	225	227	229	231
241	263	273	275	277	291	295	305	327	337	339	341	355	369	391
401	405	419	449	451	453	455	481	482						

Table 13. *x3* and *x4* corresponding to block *P* texture template (decimal).

21	25	29	49	52	53	57	73	77	81	85	105	109	113	117
137	141	145	149	157	165	169	173	177	181	201	205	209	213	229
233	273	277	281	285	292	293	301	305	329	333	337	341	357	361
365	393	397	401	405	428	429	457							

Table 14. *x3* and *x5* corresponding to block *P* texture template (decimal).

7	14	19	21	22	25	28	29	35	42	49	52	53	57	60
71	73	74	77	78	79	81	82	83	84	85	86	92	99	105
106	109	113	114	115	116	117	135	137	138	141	142	143	145	146
147	148	149	150	156	162	163	164	165	169	170	172	173	177	199
201	202	205	206	209	210	212	213	226	227	228	229	233	236	237
241	263	270	273	274	275	276	277	278	281	282	284	285	290	291
292	293	298	300	301	305	327	329	330	333	334	337	338	339	340
341	354	355	356	357	361	362	364	365	369	391	393	394	397	398
401	402	404	405	418	419	420	428	455	457	458	482	484		

Table 15. *x4* and *x5* corresponding to block *P* texture template (decimal).

21	28	29	52	53	60	77	84	85	92	109	116	117	141	148
149	156	157	164	165	172	173	181	205	212	213	228	229	233	236
276	277	284	285	292	293	300	301	305	333	340	341	348	356	357
364	365	397	404	405	420	428	429	484						

The following is a detailed description of the third case for the abovementioned block texture extension analysis. When the *X* block has adjacent points and is not marked as a valid point, one checks the table to extend the texture change between the adjacent blocks. Additionally, the corresponding method is proposed for the number of *x1–x5* edge points in the *X* block for one point, two points, and three points, respectively. Because there is no edge point above four points (inclusive) after a point in

the center and the optimal edge thinning process, the edge point above four points (inclusive) will not appear. The related method description will now be presented.

When there is a one-point edge point, there are five types of distributions, such as *x1*–*x5*, where *x1* and *x2* are adjacent to the *P* block; *x3* is adjacent to the *P*, *Q*, and *R* blocks; *x4* is adjacent to the *R* block; and *x5* is adjacent to the *R* and *S* blocks. Among these, *x1* is the *P* block texture template corresponding to Table 2; *x2* is the *P* block texture template corresponding to Table 3; and *x3* is adjacent to three blocks, which are *P*, *Q*, and *R*. Therefore, when the adjacent *P* block has adjacent point *p6* or *p7*, it corresponds to the *P* block texture template of Table 4; when the adjacent *Q* block *q7* has edge points, it corresponds to the *Q* block texture template of Table 5; and when the adjacent *R* block *r1* or *r8* has edge points, it corresponds to the *R* block texture template of Table 6. *x4* is the *R* block texture template corresponding to Table 7 and *x5* is adjacent to *R* and *S* 2 blocks. When the adjacent *R* block *r7* or *r8* has edge points, it corresponds to the *R* block texture template of Table 8; when the adjacent *S* block *s1* has edge points, it corresponds to the *S* block texture template of Table 9. During the lookup process, when the texture distribution of all adjacent blocks matches the corresponding texture template, the block texture extension analysis is completed and all required positions in the *X* block and adjacent areas are recorded, and the next step in the flowchart is followed. If this is not the case, it goes to the cross-block texture extension judgement.

When there are two edge points, they are divided into the left side of *x1*, *x2*, and *x3* and upper side of *x3*, *x4*, and *x5*, according to the distribution position of *x1*–*x5* in the *X* block, and whether the two points are on the same side is judged by the position of the edge point. When the two edge points do not fall on the same side, the method is the same as the one edge point, and the adjacent block texture templates are sequentially checked and compared; when the two edge points are located on the same side, there are three cases on the left side, comprising *x1* and *x2*, *x1* and *x3*, and *x2* and *x3*, and the corresponding *P* block texture templates are shown in Tables 10–12. When there is an edge point in *x3* and because it is adjacent to the *Q* and *R* blocks, when the *Q* block *q7* has an edge point or the *r1* or *r8* in *R* block has an edge point, one will sequentially lookup and compare the corresponding Tables 5 and 6. There are three cases of *x3* and *x4*, *x3* and *x5*, and *x4* and *x5* on the upper side, and the corresponding *R* block texture templates are shown in Tables 13–15. When there is an edge point in *x3* and because it is adjacent to the *P* and *Q* blocks, when the *P* block *p5* or *p6* has an edge point or the *Q* block *q7* has an edge point, one will sequentially look up and compare the corresponding Tables 4 and 5. When *x5* has edge points and because it is adjacent to the *S* block, it must be compared with the corresponding Table 9 when the *S* block *s1* has edge points. In the process of checking the table, when the texture distribution of the adjacent block matches the corresponding texture template, the block texture extension analysis is completed and all required positions in the *X* block and adjacent areas are recorded, and the next step in the flowchart is followed. If this is not the case, the cross-block texture extension judgement is performed.

When there are three edge points, there are two cases according to the distribution position of *x1*–*x5* in the *X* block. In the first case, when *x3* has edge points, there will be two edge points on both the left and upper sides. In the second case, when *x3* has no edge point, there are two points on one side and one point on the other side. For the case where there are two points on one side, the block texture extension analysis is performed in the manner of the above two edge points, and in the case where there is only one point on one side, the block texture extension analysis is performed in the manner of the above one edge point. Next, when the adjacent block texture extension is judged in the table comparison process, the method of judging the cross-block texture extension is further required when the non-conforming block texture template is checked. The cross-block texture extension judges that the position of the edge point of *x1*–*x5* in the *X* block is used as a judgment starting point, and searches for and extends the adjacent edge point by a range of eight adjacent points. When the adjacent edge points are less than three, and none of them are marked as a valid point, the *X* block and the adjacent edge points are not marked, and then, the block texture extension analysis is completed and the next step of the flowchart is performed. When the number of adjacent edge points is within three

(inclusive) and there are edge points marked as valid, the block texture extension analysis is completed and all required positions in the *X* block and the adjacent areas are recorded, and the next step in the flowchart is followed. When the number of adjacent edge points reaches three, the block texture extension analysis is completed and all required positions in the *X* block and the adjacent areas are recorded, and the next step in the flowchart is followed.

Two examples are used to illustrate the analysis process of adjacent block texture extension judgment and cross-block texture extension judgment. First, in the adjacent block texture extension analysis, a more complex three-point edge in *X* block *x1–x5* is used.

Figure 10 shows a schematic of the texture extension of adjacent blocks. First, one determines the number of edge points of *x1–x5* in the *X* block, and knows that in the *X* block, there are three edge points, consisting of *x1*, *x3*, and *x5*. Then, one can determine that *x1–x5* is in the *X* block according to the distribution position of the above three points in the block. Because there is an *x3* edge point, there are two edge points on both the left and upper sides, so one sequentially checks whether the adjacent blocks have adjacent points. When there are adjacent points, the adjacent block is further checked and compared.

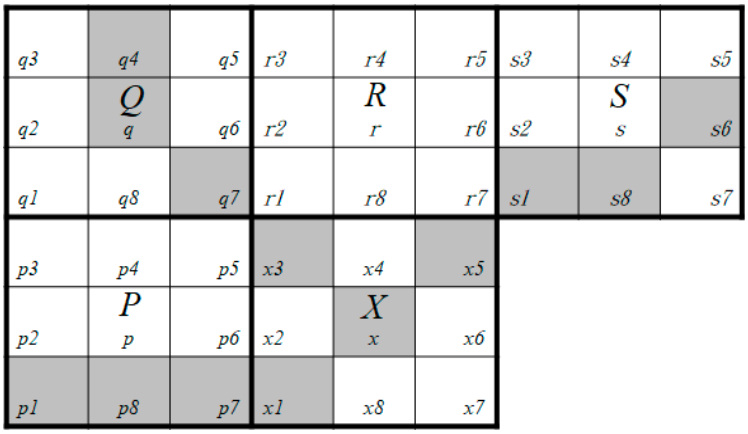


Figure 10. Schematic of texture extension of adjacent blocks.

To start the next step, one checks the edge point of the adjacent *P* block *p7*, and then compares the *P* block texture template of Table 11 with two points on the left side. Table 11 shows that the *P* block texture code is 000000111 and the decimal is 7, so one knows that the data in the first column is consistent. When the adjacent *Q* block *q7* has edge points, based on the *Q* block texture template corresponding to Table 5 to check the table comparison, it can be known that the *Q* block texture code is 010010001, the decimal is 145, and it is consistent with the third data in the fourth column in Table 5. When the block texture of the adjacent block *R* has no adjacent points, no table lookup is performed. When the adjacent *S* block s1 has edge points, it corresponds to the *S* block texture template in Table 9. From the table lookup and comparison, it can be known that the *S* block texture code is 000001110, the decimal is 14, and it is consistent with the second data of the first column in Table 9. The block texture extension analysis is then completed and the next step of the flowchart is performed after recording all required positions of edge points in the *X* and the adjacent *P*, *Q*, and *S* blocks.

Figure 11 shows the cross-block texture extension diagram.

First, one determines the number of edge points of *x1-x5* in the *X* block and finds that *x2* has one edge point. One then checks the edge points in the adjacent block and finds that there is an edge in the *P* block *p6*; the table comparison is performed with the *P* block texture template corresponding to Table 3. After checking the table, there is no matching data, and the cross-block texture extension judgment is performed. First, one searches for eight adjacent points with *x2* as the center to find *p6*. Then, one searches for eight adjacent points centered on *p6*, and finds *p4*. Finally, one searches for eight adjacent points with *p4* as the center and finds *q8*. When the search process encounters the edge point that has been marked as a valid point or when the extension of three pixels has not been marked

as a valid point, the search is stopped; that is, the block texture extension analysis is completed. The positions of edge points in the *X* block and the adjacent edge points *p6*, *p4*, and *q8* are recorded, and the next step of the flowchart is then performed.

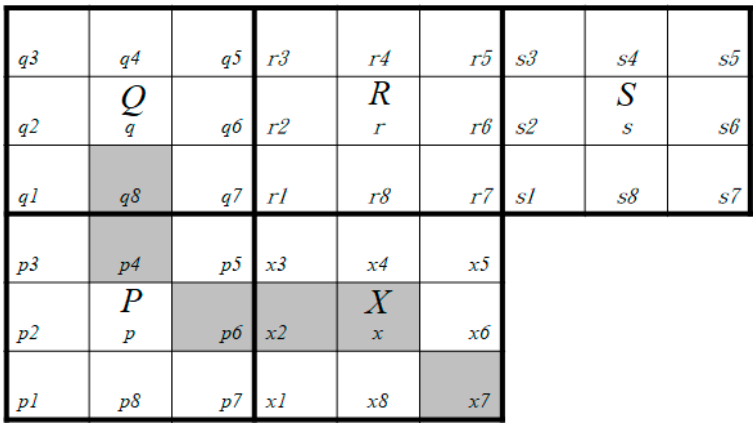
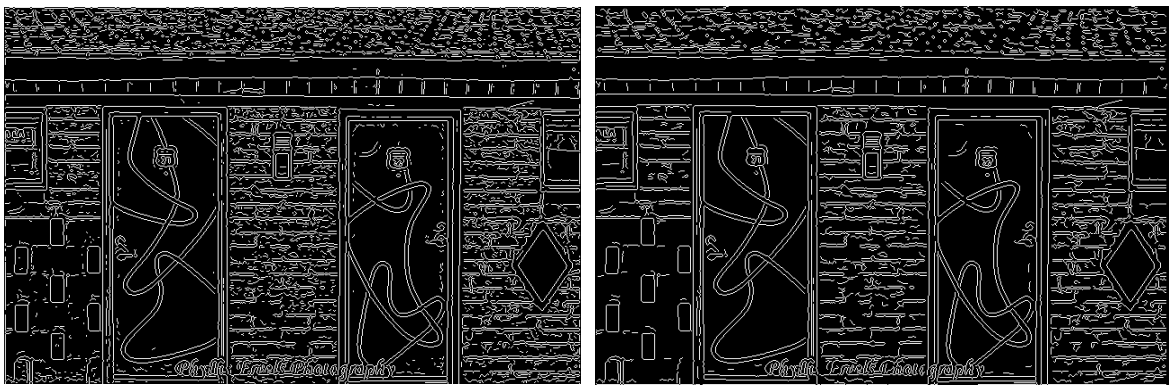


Figure 11. Cross-block texture extension diagram.

After the block texture extension analysis is completed, the judgment is made as to whether to perform the marking process. The marking process is performed according to the position of the edge point recorded in the analysis process to be marked as a valid point. The adjacent block texture extension aims to mark the edge point positions in block *X* and all adjacent blocks *P*, *Q*, *R*, and *S*. The cross-block texture extension aims to mark the edge point position in block X and extended edge points. During the marking process, the edge points that have been marked as valid are not duplicated. After all the edge points have been processed, the data marked as valid points are output; that is, the edge image after the edge texture construction process is completed. Figure 12 shows a comparison of the edge images obtained by the optimal edge thinning process and the edge texture construction processing. Figure 12a is the edge image after the optimal edge thinning process, including 38,835 points as the candidate edges. Figure 12b is the edge image obtained from Figure 12a after the edge texture construction processing is completed. After filtering through the effective edge texture condition, the total number of edges included is 32,582 points. Through the analysis of the total number of edge points, the edge texture construction processing is reduced by 6253 points compared to the optimal edge thinning process, reducing the edge points by about 16%. The edge image obtained by the above can reveal the retention of the texture through a long line segment, which can retain more meaningful edge texture information.



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

Figure 12. Comparison of edge images obtained by optimal edge thinning and edge texture construction. (**a**) Edge image with optimal edge thinning; (**b**) edge image after edge texture construction.

The proposed TCEDA achieves richer edge detection by retaining effective block edge texture changes. Figure 13 shows the effect of this algorithm on edge detection. The local magnification area shown in Figure 13a indicates that there are many obvious cracks on the wall. Although it is similar to the color of the wall, the construction of the effective edge texture can effectively detect the edge of the crack, as shown in Figure 13b.

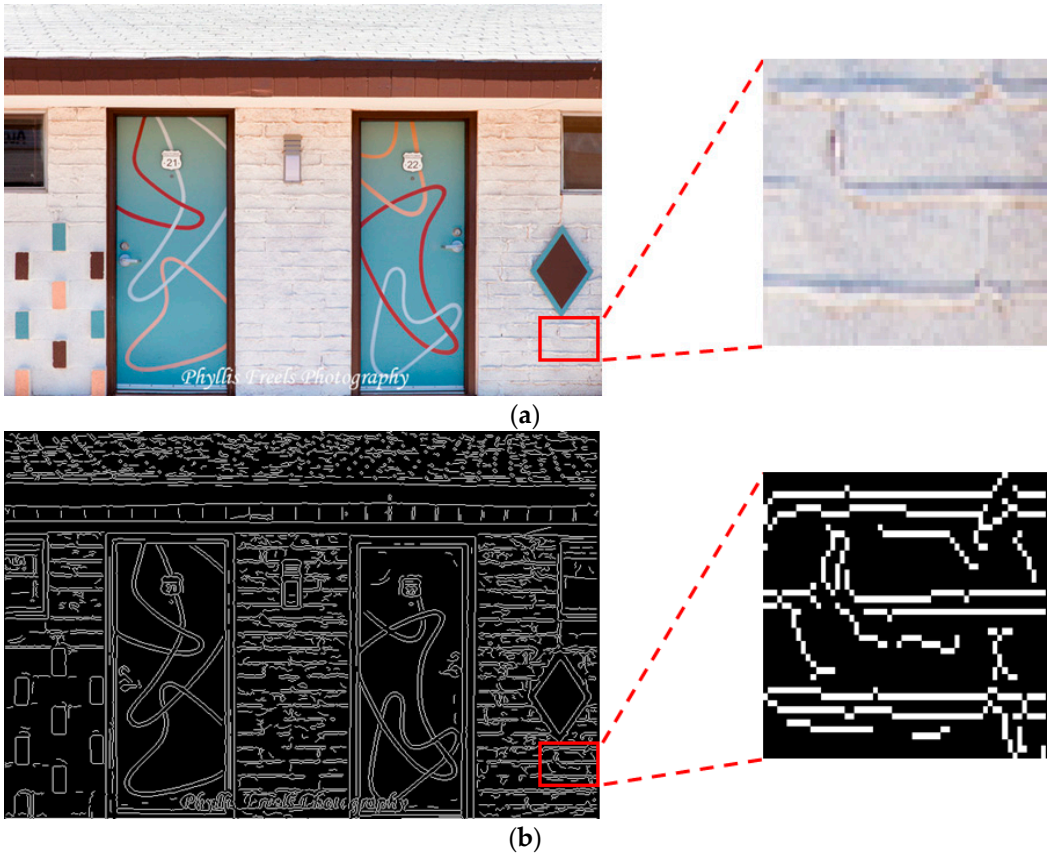


Figure 13. The effect of the proposed algorithm on edge detection. (**a**) Original image; (**b**) image after edge texture construction.

3. Experimental Results and Discussion

The TCEDA proposed in this paper was compared with the other four improved adaptive Canny edge detection algorithms. Gao and Liu [10] used the Otsu method to calculate the gradient magnitude histogram to obtain the adaptive high and low thresholds, where the low threshold was set to one-half of the high threshold; Song et al. [11] used the Otsu method to perform two operations on the gradient magnitude histogram to obtain the adaptive high and low thresholds; Saheba et al. [12] used the MSE method to perform two operations on the gradient magnitude histogram to obtain the adaptive high and low thresholds; and Li and Zhang [14] used the differential operation to calculate the gradient magnitude histogram to obtain the adaptive high and low thresholds, wherein the low threshold was set to be 0.4 times the high threshold. The TCEDA proposed in this paper and the above four adaptive Canny algorithms were implemented using Visual C++ 2015.

The source of the images tested in this study was the well-known web album Flickr [19], which is a virtual web album that provides photo-sharing and has a rich and diverse range of images. There are tens of billions of digital images stored on the website, and the number is still increasing. Through the defined keywords, the search engine can quickly find the photo you want. After multiple image tests, five images of different attributes were selected from the Flickr website (Figures 14–18) as test images and the proposed TCEDA was compared with the above four adaptive Canny algorithms analysis.

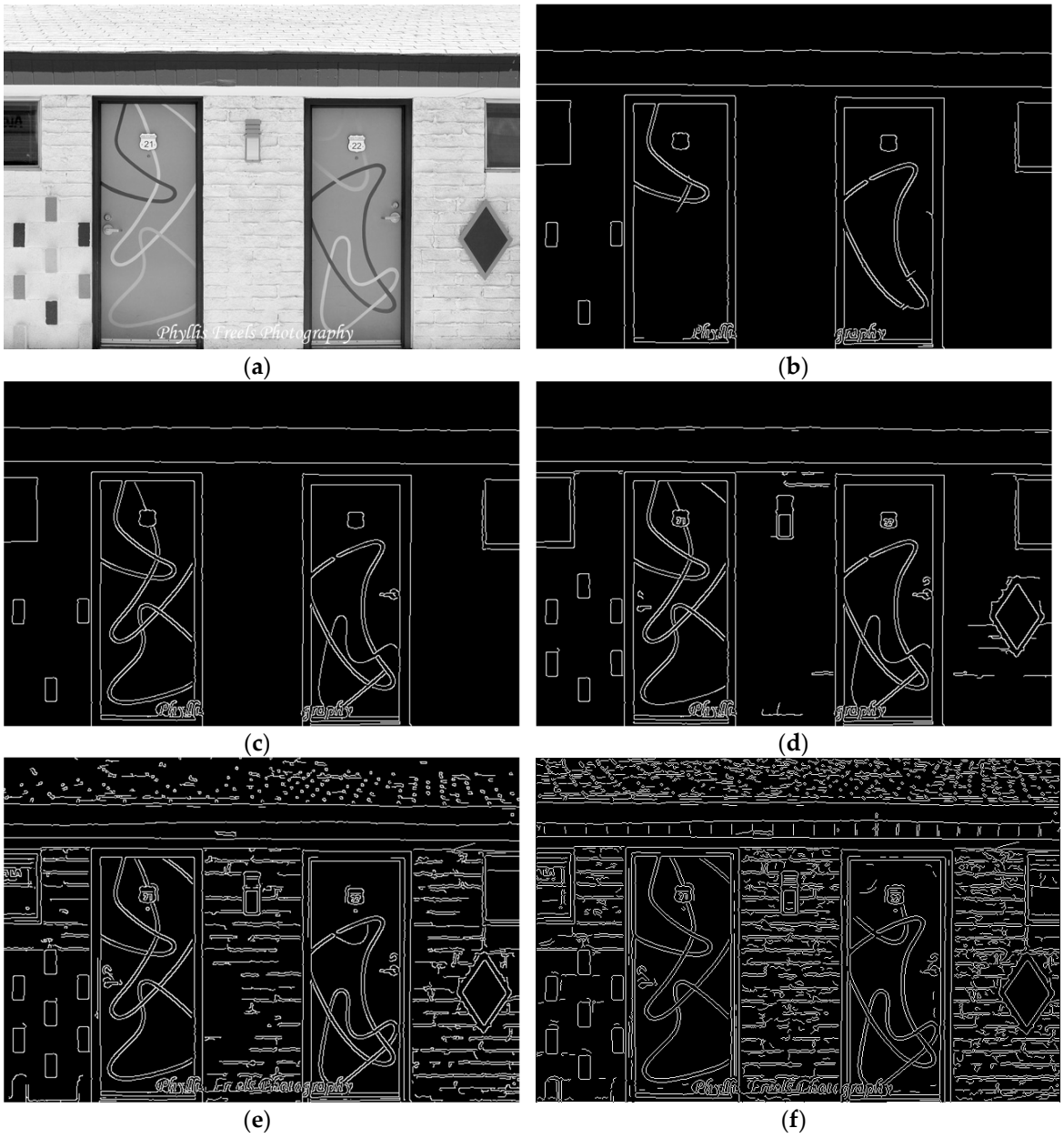


Figure 14. Hotel image comparison results of different methods. (**a**) Original image (640 × 427) [17]; (**b**) Gao and Liu's method (TH_H = 233, TH_L = 117); (**c**) Song et al. proposed method (TH_H = 233, TH_L = 38); (**d**) Saheba et al.'s method (TH_H = 141, TH_L = 41); (**e**) Li and Zhang's method (TH_H = 62, TH_L = 24); (**f**) TCEDA.

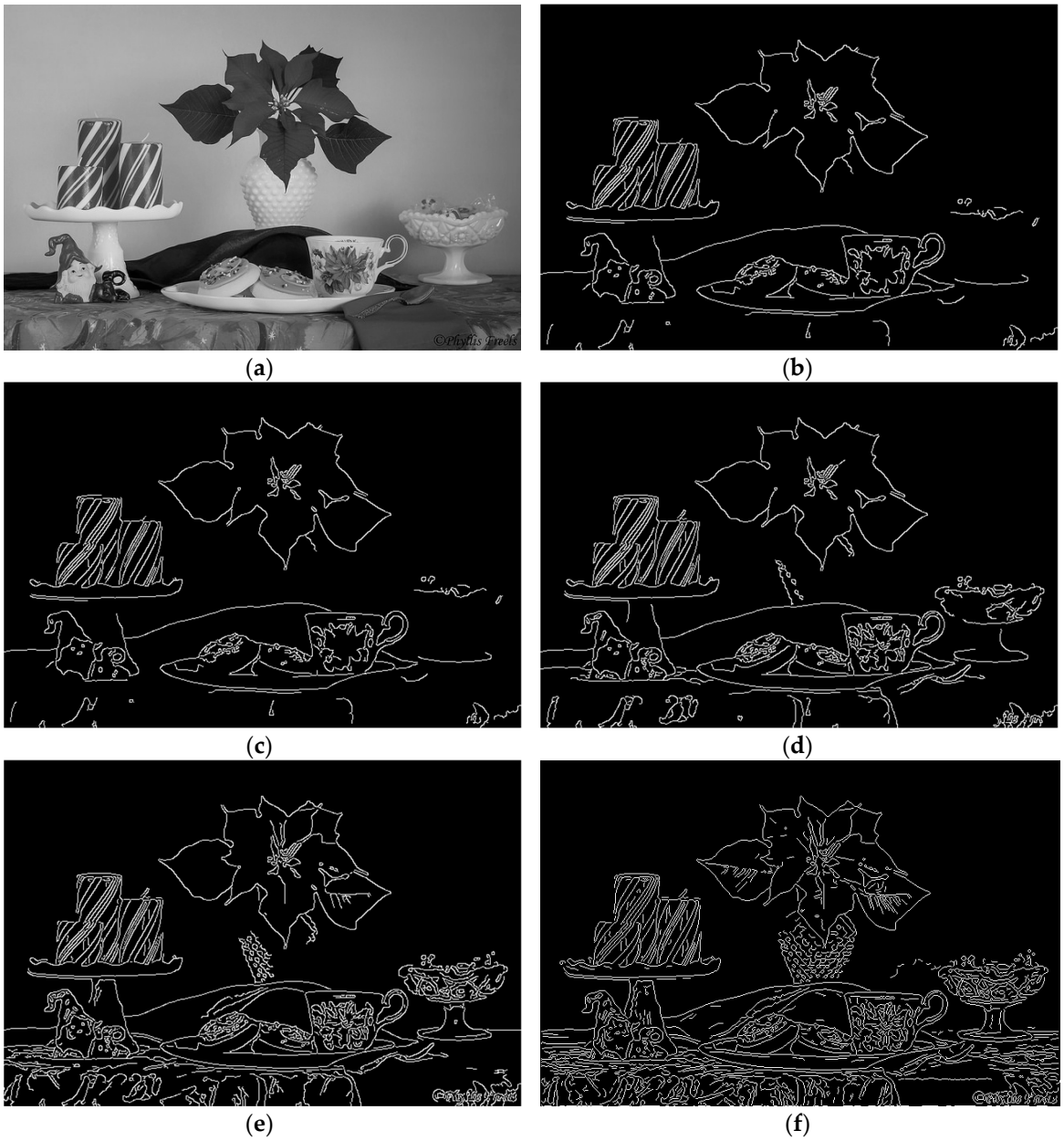


Figure 15. Glass vase image comparison results of different methods. (**a**) Original image (640 × 426) [20]; (**b**) Gao and Liu's method (TH_H = 148, TH_L = 74); (**c**) Song et al. proposed method (TH_H = 148, TH_L = 17); (**d**) Saheba et al.'s method (TH_H = 106, TH_L = 27); (**e**) Li and Zhang's method (TH_H = 65, TH_L = 26); (**f**) TCEDA.

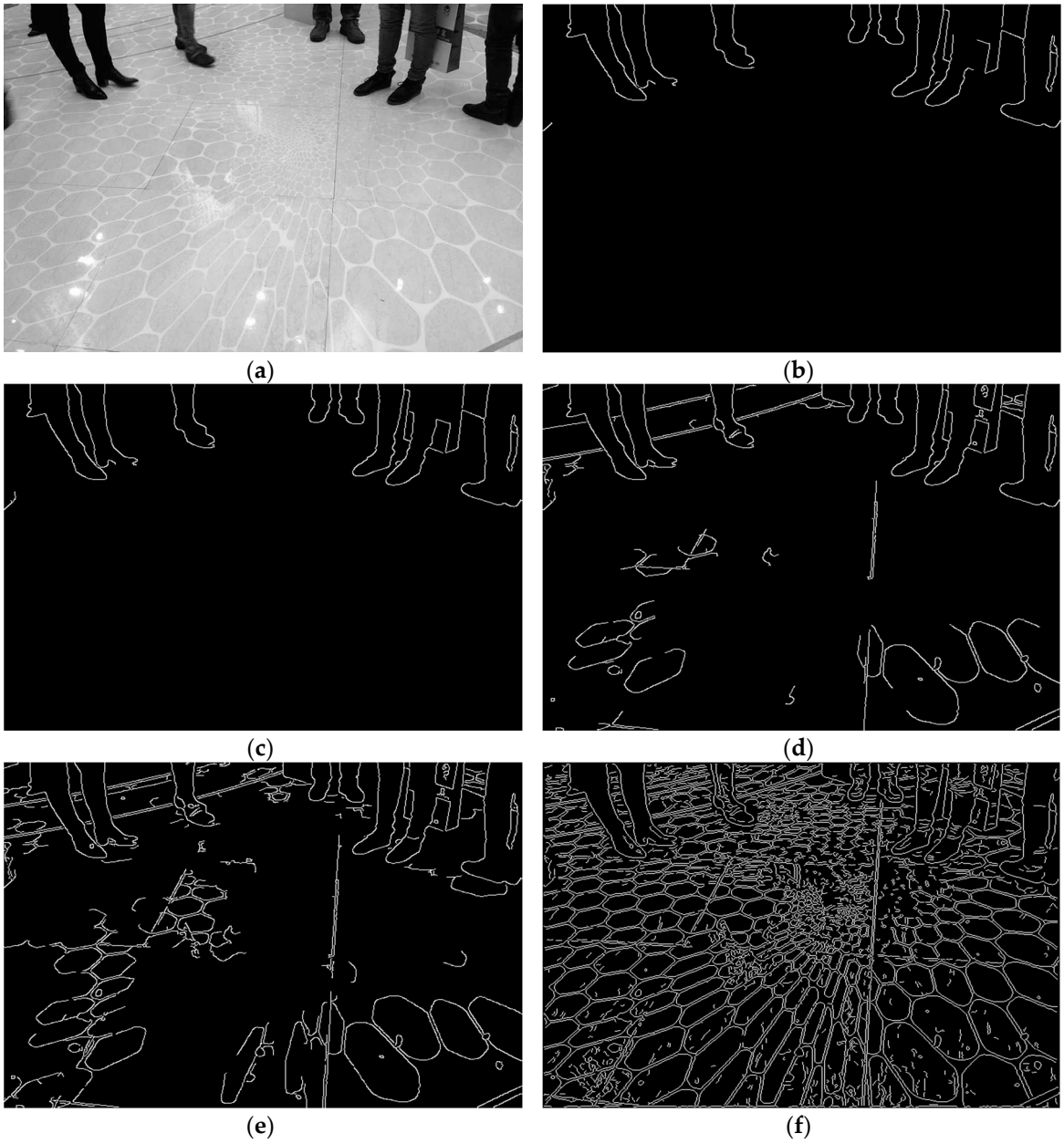
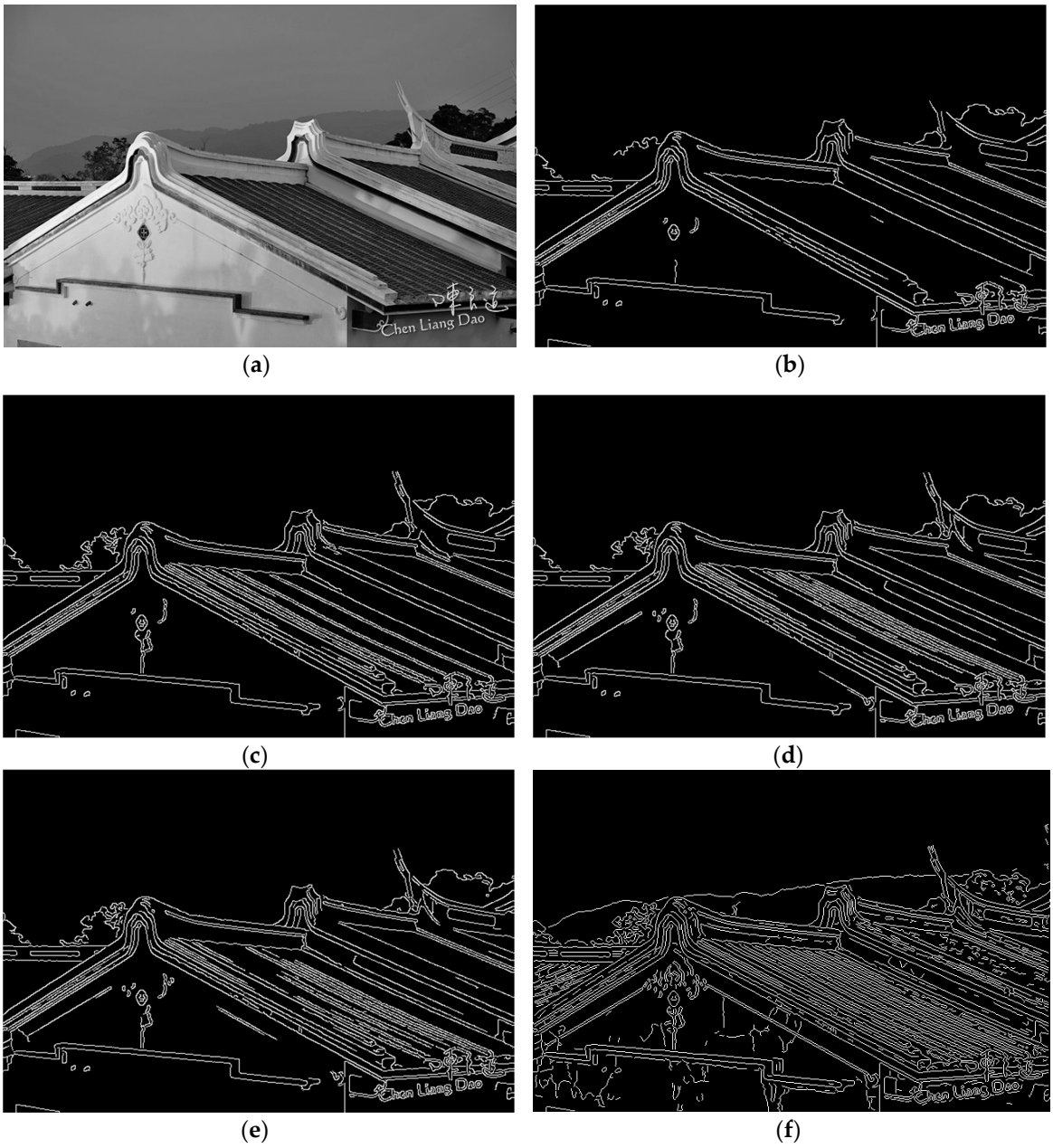


Figure 16. Embossed tile image comparison results of different methods. (**a**) Original image (640 × 480) [21]; (**b**) Gao and Liu's method (TH_H = 238, TH_L = 119); (**c**) Song et al. proposed method (TH_H = 238, TH_L = 20); (**d**) Saheba et al.'s method (TH_H = 102, TH_L = 31); (**e**) Li and Zhang's method (TH_H = 74, TH_L = 29); (**f**) TCEDA.



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

(**c**) (**d**)

(**e**) (**f**)

Figure 17. Hakka Compound image comparison results of different methods. (**a**) Original image (640 × 426) [22]; (**b**) Gao and Liu's method (TH_H = 149, TH_L = 75); (**c**) Song et al. proposed method (TH_H = 149, TH_L = 21); (**d**) Saheba et al.'s method (TH_H = 124, TH_L = 38); (**e**) Li and Zhang's method (TH_H = 112, TH_L = 44); (**f**) TCEDA.

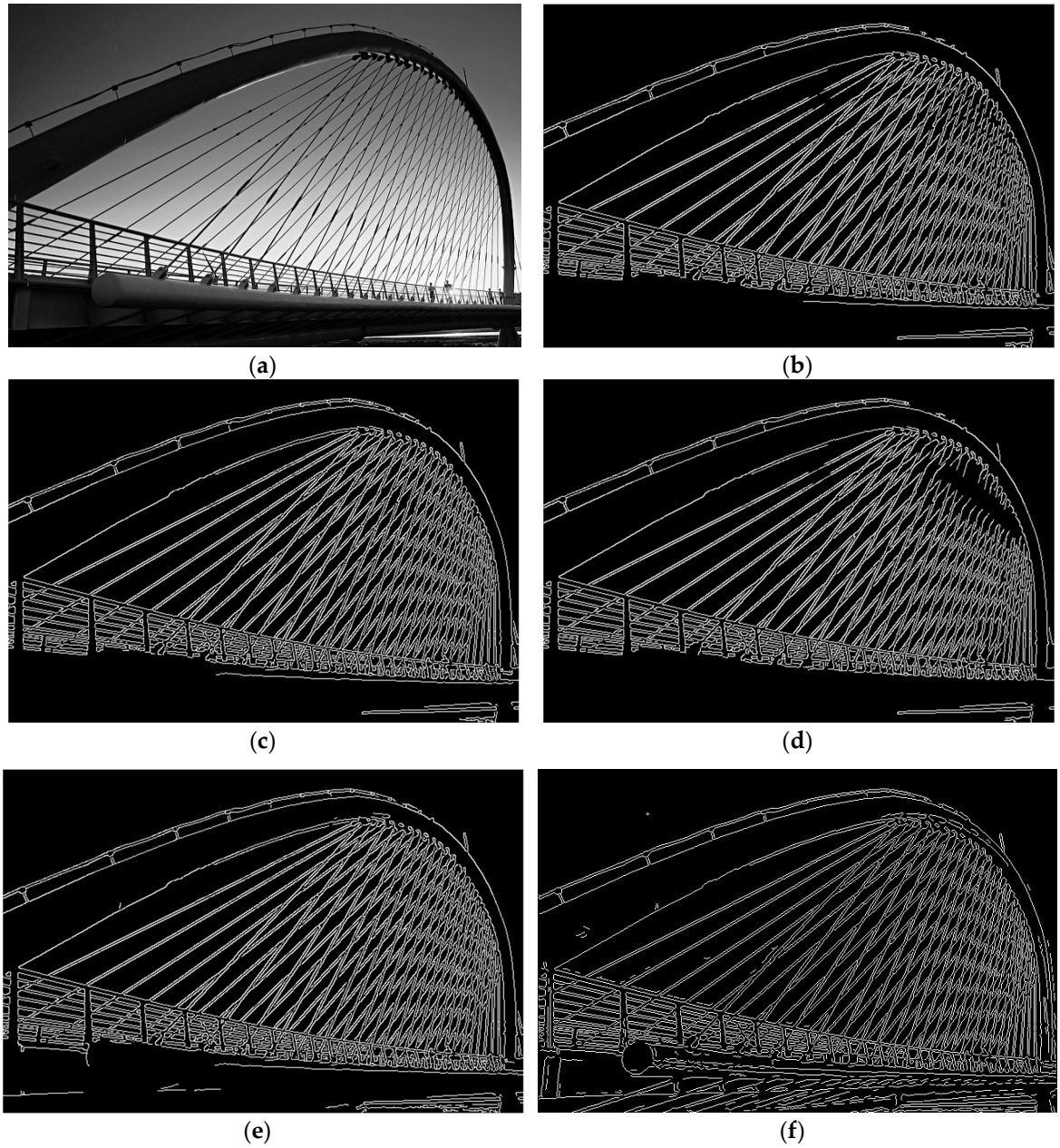


Figure 18. Bridge image comparison results of different methods. (**a**) Original image (640 × 428) [23]; (**b**) Gao and Liu's method (TH_H = 129, TH_L = 65); (**c**) Song et al. proposed method (TH_H = 129, TH_L = 11); (**d**) Saheba et al.'s method (TH_H = 154, TH_L = 55); (**e**) Li and Zhang's method (TH_H = 64, TH_L = 25); (**f**) TCEDA.

Figure 14 shows the results of a comparison of hotel images using different methods. The high-threshold TH_H and the low-threshold TH_L obtained by all methods are described below. Figure 14a shows the original test image, where there are obvious textures between the tiles laid on the roof, windows, and door panels; obvious line changes at the intersection of the door and roof; and many obvious cracks on the wall. The edge textures in the edge images from Figure 14b–d are obtained by the methods proposed by Gao and Liu, Song et al., and Saheba et al., respectively. The results of edge detection show that only some of the edge textures, such as those of the window frame, door frame, and door panel, are preserved, and many of the effective edge textures are still undetected. Figure 14e is the edge image obtained by the method proposed by Li and Zhang. The experimental results show that it retains more edge texture on the roof, door panel, and wall surface than the above

three methods, but the line changes at the intersection of the door and the roof and some of the door panels, roofs, and walls are still undetected. Figure 14f shows the TCEDA proposed in this paper. Compared to the other four methods, a significant edge texture can be effectively preserved on the roof, door panel, and wall. Besides achieving the optimal edge thinning treatment, a richer edge texture is preserved on edge detection.

Figure 15 shows the results of a comparison of glass vase images using different methods. Figure 15a is the original test image, where the background on the right side of the vase has significant shadows and there are significant texture changes on various glassware, candlesticks, napkin tablecloths, and leaves. In Figure 15c, edge images are obtained by the methods proposed by Gao and Liu and Song et al., respectively. The experimental results show that the edge detection results only preserve the peripheral contour of the leaves. The edge textures of the candlestick and some glassware still have many of the effective edge textures missing. Figure 15d shows the edge image obtained by the method proposed by Saheba et al. The experimental results show that the above two methods are preserved on the edge texture of the glassware, part of the vase, and the napkin tablecloth, but vases, leaves, and napkins still have many of the edge textures missing. Figure 15e shows an edge image obtained by the method proposed by Li and Zhang. Compared to the above three methods, it retains more edge texture on the vases, leaves, and napkins, but still cannot be more effective in terms of the edge texture detection. Figure 15f shows the TCEDA proposed in this paper. Compared to the other four methods, significant edge texture can be effectively preserved in all types of glassware, vases, candlesticks, and napkin tablecloths.

Figure 16 shows the results of a comparison of embossed tile images using different methods. Figure 16a shows the original test image with a distinct pentagonal texture above each tile, obvious seam segments between the tiles, and noticeable wrinkles and texture changes on the jeans. Figure 16b,c show the edge images obtained by the methods proposed by Gao and Liu and Song et al., and the experimental results show that the edge detection results only retain the edge contour of the foot. There are still many valid edge textures undetected. Figure 16d shows an edge image obtained by the method proposed by Saheba et al. The experimental results show that besides effectively retaining the edge contour of the foot, the pentagon texture of the partial tile is detected, but many effective edge textures are still missing. Figure 16e shows the edge image obtained by Li and Zhang's method. The experimental results show that the detection effect on the texture of the tile is better than that in the above method, but only part of the texture on the tile and the edge of the line at the seam of the tile are detected, and most of the effective edge textures are still lost. Figure 16f shows the TCEDA proposed in this paper. Compared to the other four methods, the texture on the tile, the seam line between the tiles, and the wrinkles of the jeans can be effectively preserved.

Figure 17 shows the results of comparing different methods in the Hakka Compound image. Figure 17a shows the original test image, where the tiles on the roof, the pattern on the wall with obvious light and shadow changes, and the distant skyline and antenna have obvious texture changes. Figure 17b shows the edge image obtained by the method proposed by Gao and Liu. The experimental results show that the edge detection results only preserve the edge contours of the roof and the wall periphery, and still have many effective edge textures missing. In Figure 17c–e, edge images are obtained from the experimental results by the methods proposed by Song et al., Saheba et al., and Li and Zhang, respectively. It can be seen on the roof tiles that the edges of the lines and the pattern of the wall are better than the above methods, but the edge textures, such as significant light and shadow changes on the skyline, antennas, and walls, are still not effectively preserved. Figure 17f shows the TCEDA proposed in this paper. Compared to the other four methods, the tiles on the roof, the pattern on the wall with obvious light and shadow changes, and the distant skyline and antenna are remarkable, which shows that edge textures are effectively preserved.

Figure 18 shows the results of a bridge image comparison using different methods. Figure 18a shows the original test image in which there are significant texture changes in the cable of the bridge, the railings on the bridge, and the cable supports next to the bridge. In Figure 18b–e, edge images

are obtained by the methods proposed by Gao and Liu, Song et al., Saheba et al., and Li and Zhang, respectively. The results of the test only preserve the texture changes at the cable and railings, while the edge texture of the cable bracket cannot be effectively detected. Figure 18f shows the TCEDA proposed in this paper. Compared to the other four methods, significant edge textures, such as the bridge cable, railing on the bridge, and the cable bracket besides the bridge, can be effectively preserved and further verified. The TCEDA mentioned in this paper can detect richer edge textures.

The edge detection results shown in Figures 14–18 effectively verify that the proposed TCEDA can detect the edge texture changes easily observed by the naked eye in the image and achieve richer edge detection results. The above four improved adaptive Canny edge detection algorithms use the statistics and distribution of pixel gradient values in the image to obtain the optimal threshold setting, but still lose many obvious edges in the experimental results. Therefore, the result of threshold setting directly affects the effect of edge detection. When the threshold is set too high, some edges will be lost. When the threshold is set too low, too much noise will be retained. Incorrect edge detection effects, such as incomplete object edges caused by missing edge textures or excessive false edges or noise interference, can cause erroneous results in subsequent object detection, identification, and segmentation processes.

Therefore, it can be observed from the experimental results that the proposed TCEDA can construct the edge information of the object by retaining the effective edge texture change, which effectively improves the edge loss caused by unsuitable threshold setting. There are also few parameter values to be adjusted. Compared with the traditional edge detection method, TCEDA can be widely used in various types of images.

4. Conclusions

In most edge detection methods, the problem of poor edge detection is often overcome by finding the optimal threshold value, but the meaning of the edge texture distribution in the image is neglected. Because the threshold value is based on statistics, as a result of the analysis, it is impossible to know whether important edge information in the image is effectively retained. Therefore, this paper proposes extending and constructing the edge information of the object through block analysis to analyze the change and extension characteristics of the texture in the image. The proposed algorithm effectively overcomes the phenomenon of edge loss caused by unsuitable threshold setting to get richer edge information for better results in subsequent object detection, identification, and segmentation applications.

Author Contributions: Conceptualization, S.-C.C. and C.-C.C.; Data curation, S.-C.C.; Formal analysis, S.-C.C.; Investigation, S.-C.C.; Methodology, S.-C.C. and C.-C.C.; Project administration, C.-C.C.; Software, S.-C.C.; Supervision, C.-C.C.; Validation, S.-C.C.; Visualization, S.-C.C.; Writing—original draft, S.-C.C.; Writing—review & editing, C.-C.C.

Funding: This research was funded by the Ministry of Science and Technology of Taiwan through a grant from MOST 107-2221-E-606-014-.

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

References

1. Roberts, L.G. Machine perception of three-dimensional solids. In *Optical and Electro-Optical Information Processing*; Tippett, J.T., Clapp, L.C., Eds.; MIT Press: Cambridge, MA, USA, 1965; pp. 159–197.
2. Prewitt, J.M.S. Object enhancement and extraction. In *Picture Processing and Psychopictorics*; Lipkin, B., Rosenfeld, A., Eds.; Academic Press: New York, NY, USA, 1970.
3. Duda, R.O.; Hart, P.E. *Pattern Classification and Scene Analysis*; Wiley: New York, NY, USA, 1973.
4. Bora, D.J.; Gupta, A.K. A new efficient color image segmentation approach based on combination of histogram equalization with watershed algorithm. *IJCSE* **2016**, *4*, 156–167.
5. Otsu, N. A threshold selection method from gray-level histograms. *IEEE Trans. Syst. Man Cybern.* **1979**, *9*, 62–66. [CrossRef]

6. Davis, A.M.; Arunvinodh, C.; Arathy Menon, N.P. Automatic license plate detection using vertical edge detection method. In Proceedings of the International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), Coimbatore, India, 19–20 March 2015.
7. Bradley, D.; Roth, G. Adaptive thresholding using the integral image. *J. Graph. Tools* **2007**, *12*, 13–21. [CrossRef]
8. Luo, H.B.; Jiao, A.B.; Xu, L.Y.; Shao, C.Y. Edge detection using matched filter. In Proceedings of the 27th Chinese Control and Decision Conference (CCDC), Qingdao, China, 23–25 May 2015.
9. Canny, J. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Machine Intell.* **1986**, *8*, 679–698. [CrossRef]
10. Gao, J.; Liu, N. An improved adaptive threshold canny edge detection algorithm. Proceedings of International Conference on Computer Science and Electronics Engineering, Hangzhou, China, 23–25 March 2012; pp. 164–168.
11. Song, Q.; Lin, G.; Ma, J.; Zhang, H. An edge-detection method based on adaptive canny algorithm and iterative segmentation threshold. In Proceedings of the 2nd International Conference on Control Science and Systems Engineering (ICCSSE), Singapore, Singapore, 27–29 July 2016; pp. 64–67.
12. Saheba, S.M.; Upadhyaya, T.K.; Sharma, R.K. Lunar surface crater topology generation using adaptive edge detection algorithm. *IET Image Process.* **2016**, *10*, 657–661. [CrossRef]
13. Hyndman Rob, J.; Koehler Anne, B. Another look at measures of forecast accuracy. *Int. J. Forecast.* **2006**, *22*, 679–688. [CrossRef]
14. Li, X.; Zhang, H. An improved canny edge detection algorithm. In Proceedings of the 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 24–26 November 2017; pp. 275–278.
15. Hossain, F.; Asaduzzaman, M.; Yousuf, M.A.; Rahman, M.A. Dynamic thresholding based adaptive canny edge detection. *Int. J. Comput. Appl.* **2016**, *135*, 37–41. [CrossRef]
16. Rupalatha, T.; Rajesh, G.; Nandakumar, K. Implementation of distributed canny edge detector on FPGA. *IJCES* **2013**, *3*, 22–28.
17. Phyllis Freels's Album. Available online: https://www.flickr.com/photos/phyllisfreels/9452430929/ (accessed on 3 January 2019).
18. Bottenbruch, H. Structure and use of ALGOL 60. *J. ACM* **1962**, *9*, 161–211. [CrossRef]
19. Flickr. Available online: https://www.flickr.com/ (accessed on 5 January 2019).
20. Phyllis Freels's Album. Available online: https://www.flickr.com/photos/phyllisfreels/31524363831/ (accessed on 15 January 2019).
21. Oung Dustin's Album. Available online: https://www.flickr.com/photos/idostone/15986200330/ (accessed on 15 January 2019).
22. Chen Liangdao's Album. Available online: https://www.flickr.com/photos/idisdao/7005684840/ (accessed on 15 January 2019).
23. Trannan's Album. Available online: https://www.flickr.com/photos/trannan/2969812013/in/album-72157607992888205/ (accessed on 15 January 2019).