PAPER

Enhancing Salt-and-Pepper Noise Removal in Binary Images of Engineering Drawing

Hasan S. M. AL-KHAFFAF^{†a)}, Member, Abdullah Z. TALIB^{†b)}, and Rosalina Abdul SALAM^{†c)}, Nonmembers

SUMMARY Noise removal in engineering drawing is an important operation performed before other image analysis tasks. Many algorithms have been developed to remove salt-and-pepper noise from document images. Cleaning algorithms should remove noise while keeping the real part of the image unchanged. Some algorithms have disadvantages in cleaning operation that leads to removing of weak features such as short thin lines. Others leave the image with hairy noise attached to image objects. In this article a noise removal procedure called TrackAndMayDel (TAMD) is developed to enhance the noise removal of salt-and-pepper noise in binary images of engineering drawings. The procedure could be integrated with third party algorithms' logic to enhance their ability to remove noise by investigating the structure of pixels that are part of weak features. It can be integrated with other algorithms as a post-processing step to remove noise remaining in the image such as hairy noise attached with graphical elements. An algorithm is proposed by incorporating TAMD in a third party algorithm. Real scanned images from GREC'03 contest are used in the experiment. The images are corrupted by salt-and-pepper noise at 10%, 15%, and 20% levels. An objective performance measure that correlates with human vision as well as MSE and PSNR are used in this experiment. Performance evaluation of the introduced algorithm shows better-quality images compared to other algorithms.

key words: salt/pepper noise removal, impulsive noise, engineering drawing, binary image, document analysis and recognition

1. Introduction

Noise appears in scanned engineering drawings (ED) for many reasons, such as bandwidth limitation of scanners, unequal illumination of paper documents, tape on paper drawings, and dirt [1]. Scanned EDs are binary images that contain line-like graphics as well as texts where they appear in any orientation and with different thicknesses. Therefore one-pixel-thick objects may exist in EDs. The graphical elements and the background pixels are represented by ON pixels and OFF pixels respectively. The noise may occur as OFF pixels in ON regions (called salt noise) or as ON pixels in OFF regions (called pepper noise). Reducing noise before further analysis (such as converting from raster to vector representation) of scanned images is important for the correct detection of image features. Noise removal algorithms should remove noise while preserving the other useful features of the image. Scanned EDs may contain one-pixel-wide blobs as part of the graphical elements or as

noise. Some noise removal algorithms that we have studied cannot recognize one-pixel-wide blob as a graphical object. Therefore, it is removed together with other noise. Removing graphical elements in EDs causes miss-detection during vectorization and the EditCost Index of the vectorised EDs increases accordingly [2], [3].

Retaining short noise blobs connected to graphical elements may cause difficulties for other stages of image analysis. These short blobs may mislead thinning algorithms and can end up as artificial limbs. A procedure named TAMD is developed to enhance the noise-removal operation. The procedure studies the structure of thin lines before deciding to remove or retain the thin lines. An algorithm that incorporates TAMD is presented in this article, and it is based on a single iteration filter named Enhanced kFill [4]. The proposed algorithm can eliminate salt-and-pepper noise and one-pixel-wide noise, while preserving thin graphical elements.

In the next section, a survey of different noise removal algorithms is presented. Sections 3 and 4 describe respectively the proposed noise removal algorithm and the proposed procedure (TAMD) in detail. Distortion measurements used in the performance evaluation are also described in this paper in Sect. 5. Section 6 illustrates experiment setup, presents the results of the experiment, and discussion of the performance of the proposed algorithm compared to other algorithms. Section 7 presents the conclusion.

2. Noise Removal Algorithms

2.1 Median

Median filter is a non linear filter that can remove salt-and-pepper noise from images. It has good capability of removing impulsive noise [5], but it has the disadvantage of removing fine details and sharp corners in the image [6]. The median of the set of values that include the neighboring pixels as well as the core (center) pixel of a $k * k$ window is used as the new value of the window center. The value of k is an odd number that represents the width and height of the moving window. If $X_{(i,j)}$ is the pixel to be processed then the new value is given by:

$$Y_{(i,j)} = \text{median} \{X_{(i-s,j-t)} | (s,t) \in W\} \quad (1)$$

where $Y_{(i,j)}$ is the new output value and W is the window and its coordinates is defined relative to the center. If a $3 * 3$ window is used, W is given by:

Manuscript received August 4, 2008.

Manuscript revised December 1, 2008.

[†]The authors are with School of Computer Sciences/Universiti Sains Malaysia, Malaysia.

a) E-mail: hasan@cs.usm.my

b) E-mail: azht@cs.usm.my

c) E-mail: rosalina@cs.usm.my

DOI: 10.1587/transinf.E92.D.689

$$W = \{(s, t) | -1 \leq s \leq +1, -1 \leq t \leq +1\}$$

2.2 Center Weighted Median (CWM)

An extension of the Median filter is proposed by [6] to get a better trade off between removing impulsive noise and keeping the fine details intact. The CWM gives more weight to the core pixel than to the neighboring pixels. Therefore it can control the behavior of the noise removal process by setting a suitable weight for the center.

Let's assume that $X_{(i,j)}$ is the input pixel to be processed and $Y_{(i,j)}$ is the output of the CWM. $Y_{(i,j)}$ is defined as:

$$Y_{(i,j)} = \text{median}\{X_{(i-s,j-t)}, w_c \diamond X_{(i,j)} | (s, t) \in W, (s, t) \neq (0, 0)\} \quad (2)$$

where w_c is the center weight and $\diamond$ denotes the replication operator. For a 3×3 window, W can be given by:

$$W = \{(s, t) | -1 \leq s \leq +1, -1 \leq t \leq +1\}$$

2.3 Morphological Operators

Morphological operators could be used to remove noise in binary images. They are used to analyze images by means of set theory [4], [5]. Dilation and erosion are two operators used for removing noise. In dilation the border of the graphical objects is grown by adding another layer of ON pixel to it. In erosion, the graphical elements are shrunk by removing one layer of ON pixels from its boundary. Dilation and erosion are combined together to produce other operators namely closing and opening. Closing operator (dilation followed by erosion) is used to remove salt noise while opening (erosion followed by dilation) removes pepper noise. Removing of salt-and-pepper noise by these filters requires a sequence of opening-closing operations. The disadvantages of using opening and closing operators are removal of thin lines and the joining of very close objects respectively.

2.4 kFill

kFill algorithm [7], [8] could remove salt-and-pepper noise from binary images. The $k \times k$ window is moved in a raster scan manner over the image and the core of the window is set to ON or OFF depending on three variables: n , c , and r . There are two subiterations for ON and OFF filling. For ON (OFF) filling subiteration, the variable n represents the number of ON (OFF) pixels in the window, the variable c is the number of connected components with ON (OFF) pixels inside the window, and the variable r is the number of ON (OFF) pixels in window corners. The core is excluded from calculation of the three variables. The condition for filling is given by:

$$(c = 1) \wedge [(n > 3k - 4) \vee [(n = 3k - 4) \wedge (r = 2)]] \quad (3)$$

The disadvantage of this algorithm is the shortening of one-pixel-wide graphical objects from their end points.

2.5 Enhanced kFill

Another algorithm based on kFill as proposed in [4] tries to bypass the disadvantages of kFill. In this article it is called Enhanced kFill. The rules are modified in order to be able to remove noise spots smaller than windows' core size. Instead of having two separate subiterations for ON and OFF filling, the number of ON pixels inside the core and Eq. (3) will be used to decide whether to perform ON or OFF filling. Hence the algorithm can perform in one iteration. Flowchart of this algorithm is shown in Fig. 1.

Its disadvantages are: (i) with small value of k (3 for example) the algorithm only shortens one-pixel-wide blobs connected to graphical elements without removing it while it also shortens one-pixel-wide graphical elements, (ii) with larger value of k (4 and above) some graphical elements are extensively eroded.

2.6 Activity Detector

In a more recent study by [1], a noise removal algorithm based on activity detection (hence called Activity Detector) is proposed. The algorithm can remove salt-and-pepper noise in electronic documents that contain texts, dithered patterns, and graphics. First, all connected components (CC) are computed with their bounding boxes (in our implementation we use the method proposed by [9] to compute CCs). Each CC bounding box is expanded and will participate to the calculation of activity map by the value of one for each of its pixels. Next the CCs are classified into three sets depending on their activity measure and number of pixels. The first set will include CCs near text areas or dithered areas. The second set includes only CCs in dithered areas. The rest of CCs belongs to the third set which includes areas of graphical elements. Finally the salt-and-pepper noise removal is performed by removing selected CCs that follow prescribed rules. The disadvantage of this method is in the

Fig. 1 Flowchart of enhanced kFill. Taken from [4].

use of many parameters (five). Another issue is related to the removal of noise connected to graphical elements. Since the removal of noise is implemented by removing the whole CC, it cannot remove such noise as it is considered part of that CC.

3. Proposed Noise Removal Algorithm

It is shown that the above mentioned methods are not reliable in processing the structure of thin graphical elements and causes distortions that are difficult to recover (when a line is totally removed, for example) in the next phases of image analysis. A noise removal algorithm (based on Enhanced kFill) is proposed in this section to overcome some of the weaknesses in noise removal algorithms. A procedure named TrackAndMayDel short named TAMD (explained in detail in Sect. 4) is also proposed to investigate the structure of thin lines before changing pixels values. This procedure is used by the proposed noise removal algorithm to help decide pixel values when the information inside the 3×3 window is not sufficient.

Before describing the proposed noise removal algorithm, the following definitions are set: Let I be a binary image with B and F as the set of background and foreground pixels, respectively. Foreground and background pixels have the value of 1 (or ON) and the value of 0 (or OFF), respectively. Note that $B \cap F = \phi$. The negation operator is represented by a bar hence $\bar{B} = F$, and $\bar{F} = B$ will hold. Every pixel p (either in F or in B) may have up to eight direct neighbors (or 8-neighbors) as shown in Fig. 2.

An end point is a pixel which has only one 8-connected neighbor pixel similar to its value. A branch point is a pixel which has four or more neighbors with the same intensity. The size of the window used through noise removal and tracking operations is 3×3 pixels. The pixel at the center of the window is called the core.

The algorithm performs a single horizontal scan over the image with no subiterations. The value of the core pixel, either OFF or ON determines whether the core is a candidate for ON or OFF fill, respectively. The flowchart is as shown in Fig. 3.

To simplify the description, it is explained in the context of ON fill (i.e. when the value of the core is OFF). The OFF fill is similar, one needs to replace every occurrence of ON (in the rest of this section) with OFF. The boundary of the core is inspected to decide whether it has potential to be filled. Three variables, c , n and r are computed from the neighborhood, where c is the number of 8-connected components with ON, n is the number of ON pixels and r is the number of corner pixels of value ON. If Eq. (3) is satisfied, either Eq. (4) or Eq. (5) should then hold. If conditions in Eq. (4) are met:

$$(c = 1) \wedge [(n = 6) \vee (n = 8) \vee [(n = 5) \wedge (r = 2)]] \quad (4)$$

$$(c = 1) \wedge (n = 7) \wedge [(r = 3) \vee (r = 4)] \quad (5)$$

The core is filled with ON. Otherwise if the conditions

p[7]	p[0]	p[1]
p[6]	p	p[2]
p[5]	p[4]	p[3]

Fig. 2 The eight neighbors of pixel p .

Fig. 3 Flowchart of proposed noise removal algorithm.

Fig. 4 Sample window configurations during ON fill subiteration.

in Eq. (5) are met (i.e. core is an end point), then no sufficient information is available to decide the proper action and hence TAMD will be called to investigate the structure of line's pixels connected to the core. Figure 4 shows examples for different window contents during ON fill subiteration. Values for the three variables are also shown. Note that in the original algorithm the core is directly filled with ON if the conditions in Eq. (3) are met. Equation (3) in the original algorithm is transformed into two Eqs. (4) and (5) in our proposed algorithm; the core will be filled with ON if Eq. (4) is met. Otherwise TAMD is called which may/may not change the core. Note also that Eq. (5) can be simplified to $(n = 7)$. The simplification is trivial and needs no derivation. However, from Fig. 4-c it can be shown that if the pixel with value 0 (currently in $p[5]$) is relocated to any of the other seven places inside the window, Eq. (5) will hold and at the same time $n = 7$ will also hold. TAMD which is responsible for investigating thin lines is explained in the next section.

4. Proposed TrackAndMayDel Procedure (TAMD)

The idea of the proposed TAMD procedure is to perform fur-

ther investigation before changing end points. The method starts performing when an end point is detected (i.e. when Eq. (5) evaluates to true). The purpose of this procedure is to check whether the already detected end point is part of thin line or part of small spurious branch connected to a graphical element. The TAMD procedure gives the proposed noise removal algorithms (described in Sect. 3) the ability to check pixels outside the 3×3 window. This widens the area of the image viewed by the algorithms without the need to use a larger window size (which is costly in term of processing time). An algorithmic description and flowchart are shown in Fig. 5 and Fig. 6 respectively. When there are two actions inside the process symbol in Fig. 6, the first one ends with ‘.’ in order to separate the two actions.

In this paragraph the mathematical foundation for TAMD is presented. As in Sect. 3, ON fill subiteration is assumed in the rest of this section. For OFF fill the process is similar. One only needs to remove the negation operator from Eqs. (6) and (7) derived below. The definitions set in Sect. 3 need to be recalled. The distance between two pixels is denoted by $d(.,.)$. For a set of connected pixels $p_0, p_1, \dots, p_n$ the distance $d(p_0, p_n) = n$ if p_i is 8-neighbor of p_{i+1} for all $0 \leq i < n$. Informally, $d(p_0, p_n)$ is the number of steps need to be performed in order to traverse from p_0 to p_n passing through all the pixels in between. The cardinality of a set is denoted by $|\cdot|$.

Number of neighbors: Denoted as $NN(p)$ is the number of 8-neighbors for pixel p and it is defined by:

$$NN(p) = \sum_{k=0}^7 \overline{p[k]} \quad (6)$$

neighbor Set: Denoted as $NS(p)$ is the set of 8-neighbors for pixel p and it is given by:

$$NS(p) = \{q \mid d(p, q) = 1, q \in \bar{F}\} \quad (7)$$

```

TrackAndMayDel (x, y, Flag)
{
  // Input:
  // x, y is the coordinate of the core pixel which should be an end point.
  // Flag is the type of filling operation (either ON or OFF).
  // Action taken:
  // A chain of 8-connected pixels of length no more than LT starting from (x, y)
  // are either changed (filled) or retained
  Define A as a set of candidate pixels to be changed. Initialize set A to empty
  Add coordinates (x, y) to A
  C = 1 // Number of entries in the set A
  NN = 2 // Number of neighbor pixels of value !Flag. Initially it is set to 2
  DIR = Direction from pixel (x, y) to its neighbor pixel of value !Flag
  while (C < LT) && [(NN == 2) || (NN == 3)] do // Tracking
  {
    (x', y') = GetNeighborPixel (x, y, DIR)
    DIR' = Real direction from pixel (x, y) to its neighbor pixel (x', y')
    NN = Number of neighbor pixels of (x', y') with value !Flag
    if (NN <= 3)
    {
      Add coordinates (x', y') to A
      C = C + 1
    }
    (x, y) = (x', y')
    DIR = DIR'
  }
  if [(NN > 3) || (NN == 1)] && (C <= LT) // Decision making
  {
    Change pixels value in input image whose coordinate in A to value of Flag
  }
  Set A to empty
}

```

Fig. 5 Proposed TAMD algorithm.

The first stage of TAMD is to create the set A which contains a connected set of pixels belonging to a short line. Set A is formulated as below:

$$A = \{p_i \mid NN(p_0) = 1, p_{i+1} \in NS(p_i) \forall 0 \leq i < n, n \leq LT\} \quad (8)$$

where LT is a predefined threshold representing the maximum number of pixels to investigate by TAMD.

Fig. 6 TAMD flowchart.

Fig. 7 Four different cases where TAMD is in operation.

Fig. 8 Test images used in our experiment.

The final decision to retain or remove the pixels in A is by relying on the following equation:

$$(|A| \leq LT) \wedge [(NN(p_n) = 1) \vee (NN(p_n) > 3)] \quad (9)$$

If Eq. (9) is turned out to be true, all pixels $p \in A$ will be removed otherwise they will be retained.

The two parts of the procedure is explained next. First a tracking part in which the branch is tracked starting from its end point and the tracked pixels are recorded. Tracking will stop when either an end point or branch point is reached; or

Table 1 Ratio of foreground pixels to the total number of image pixels for the test images.

Image	Foreground	Total	Ratio
1.tif	10,215	169,280	6.0%
2.tif	8,569	150,880	5.6%
2_100.tif	4,936	150,880	3.2%
3.tif	8,798	150,144	5.8%
3_100.tif	4,579	150,144	3.0%
4.tif	6,030	72,576	8.3%

(a) Image 1.tif

(b) Image 2.tif

(c) Image 2_100.tif

(d) Image 3.tif

(e) Image 3_100.tif

(f) Image 4.tif

Fig. 9 PSNR for different algorithms on many images corrupted by 10% salt-and-pepper noise.

(a) Image 1.tif

(b) Image 2.tif

(c) Image 2_100.tif

(d) Image 3.tif

(e) Image 3_100.tif

(f) Image 4.tif

Fig. 10 PSNR for different algorithms on many images corrupted by 15% salt-and-pepper noise.

(a) Image 1.tif

(b) Image 2.tif

(c) Image 2_100.tif

(d) Image 3.tif

(e) Image 3_100.tif

(f) Image 4.tif

Fig. 11 PSNR for different algorithms on many images corrupted by 20% salt-and-pepper noise.

(a) Image 1.tif

(b) Image 2.tif

(c) Image 2_100.tif

(d) Image 3.tif

(e) Image 3_100.tif

(f) Image 4.tif

Fig. 12 DRD for different algorithms on many images corrupted by 10% salt-and-pepper noise.

Fig. 13 DRD for different algorithms on many images corrupted by 15% salt-and-pepper noise.

the number of tracked points exceeds a predefined threshold (LT). Second, the procedure will remove the already tracked branch if it is shorter than or equal to LT ; and it is either connected to thick element or it is a separate object. Tracking is performed by means of GetNeighborPixel procedure (the 4th step in Fig. 6). The functionality of this procedure is described by means of an example. Suppose tracking starts at pixel (x, y) . Finding the neighbor of pixel (x, y) involves searching its neighborhood. We follow a trend that makes it easier and faster to get the neighbor by searching in a suggested direction first, and then in the directions that makes little deviation with it. For the very first pixel (x, y) , there is no suggested direction. So, the search starts in its neighbors one by one till it finds the only neighbor. Lets call it (x', y') . The real direction from (x, y) to (x', y') is considered as the suggested direction to be used in tracking from (x', y') to its neighbor. The operation continues until stop conditions are met. Figure 7 shows an example of four different cases where TAMD is in operation. The value of LT is assumed to be 4 in this example. Figures 7(a) and (b) show segments that will be eliminated while Figs. 7(c) and (d) show segments that will be retained. The pixels that are already tracked are shown in grey. Other parts of the graphical elements are shown in black, and the background is white. The pixel where tracking starts is shown as grey square with a filled black circle inside. The images are scaled up to make it easier to visualize.

5. Distortion Measurements

5.1 Mean Square Error (MSE) and Peak Signal-to-Noise Ratio (PSNR)

The mean-square-error (MSE) and peak signal-to-noise ratio (PSNR) are the traditional criteria used for evaluating image processing algorithms [5], [10], [11]. The calculation of these terms are based on the amount of changed pixels between the original and the processed image. For a source image $S_{(i,j)}$ and processed image $Y_{(i,j)}$, MSE is given by:

$$MSE = \frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N (S_{(i,j)} - Y_{(i,j)})^2 \quad (10)$$

$PSNR$ which is measured in dB is given by:

$$PSNR = 20 \cdot \log_{10} \left(\frac{Max}{\sqrt{MSE}} \right) \quad (11)$$

where Max is the maximum possible intensity. For one bit per pixel images $Max = 1$. Elements in binary document images (whether characters, symbols, or lines) are composed of sets of connected pixels with the same intensity (usually black) bounded by other sets of connected pixels (usually white). Distance among pixels is an important feature in this type of documents and the pixels of the same intensity are tightly related to their neighboring pixels. For example, a white pixel that changed to black is more noticeable to the human eyes when all its neighbors are white

Fig. 14 DRD for different algorithms on many images corrupted by 20% salt-and-pepper noise.

while it is less detectable when many of its neighbors are black. The neighborhood of the changed pixels will not contribute to the calculation of MSE and PSNR. Therefore it will not correlate with human vision system in the case of document images [10], [11].

5.2 Distance Reciprocal Distortion Measurement (DRDM)

An objective measure for distortion in document images is proposed in [10], [11]. This method which is called distance-reciprocal distortion measure (DRDM) is developed specially for document images where the graphical elements are separated from the background by clear edges. It takes advantage of human visual system, that is, the change of a pixel is more visible when the focus of vision is on its neighbor. The sensitivity of vision to flipping the pixel is increased when the two pixels are close to each other. This measure is based on creating a weighted matrix with weights computed as the reciprocal distance from each pixel to the center pixel within a small neighborhood of $m \times m$ window size. A brief description of this method is shown next. For an image $f(x, y)$ and a processed image $g(x, y)$ the weighted matrix is denoted as W_m with size $m \times m$, $m = 3, 5, 7, \dots$. The center of the matrix is at (i_c, j_c) , $i_c = j_c = (m + 1)/2$. $W_m(i, j)$ is defined as:

$$W_m(i, j) = \begin{cases} 0, & \text{for } i = i_c \text{ and } j = j_c \\ \frac{1}{\sqrt{(i - i_c)^2 + (j - j_c)^2}} & \text{otherwise} \end{cases} \quad (12)$$

The matrix should then be normalized as shown below:

$$W_{Nm}(i, j) = \frac{W_m(i, j)}{\sum_{i=1}^m \sum_{j=1}^m W_m(i, j)} \quad (13)$$

If S pixels are changed in $g(x, y)$, distance-reciprocal distortion denoted as DRD_k should be calculated for each flipped pixel. For the k th flipped pixel at $(x, y)_k$ in the processed image $g(x, y)$ the distortion is calculated from block B_k in $f(x, y)$ that is centered at $(x, y)_k$. The distortion of the k th pixel is given below:

$$DRD_k = \sum_{i=1}^m \sum_{j=1}^m [D_k(i, j) \times W_{Nm}(i, j)] \quad (14)$$

where $D_k(i, j)$ is the (i, j) member of the difference matrix and it is given by:

$$D_k(i, j) = |B_k(i, j) - g[(x, y)_k]| \quad (15)$$

DRD_k is the weighted sum of the pixels in the block B_k . Finally, the DRD of the processed image $g(x, y)$ is given below:

$$DRD = \frac{\sum_{k=1}^S DRD_k}{NUBN} \quad (16)$$

where $NUBN$ is the number of nonempty area in the image. It is taken as the number of non uniform blocks of 8×8 pixels in $f(x, y)$.

Table 2 Performance comparison among different algorithms (10% noise level).

Image	Criteria	OpenClose	kFill	Enh kFill	ActDetec	Median	CWM 3	CWM 5
1.tif	PSNR	16.58	22.33	22.36	21.68	19.35	21.24	21.37
	MSE	0.022	0.0058	0.0058	0.0068	0.0116	0.0075	0.0073
	DRD	2.4764	0.9273	0.9223	0.9649	1.1128	0.782	1.287
2.tif	PSNR	16.18	21.99	22.11	21.71	20.56	22.3	21.09
	MSE	0.0241	0.0063	0.0062	0.0067	0.0088	0.0059	0.0078
	DRD	2.931	1.1056	1.0743	1.0959	0.9825	0.7367	1.4706
2_100.tif	PSNR	16.27	21.16	21.35	21.27	18.43	19.91	20.85
	MSE	0.0236	0.0076	0.0073	0.0075	0.0144	0.0102	0.0082
	DRD	2.3953	1.2003	1.1503	0.9778	1.2848	0.9502	1.4782
3.tif	PSNR	15.92	22.35	22.5	22.17	20.92	23.13	21.69
	MSE	0.0256	0.0058	0.0056	0.0061	0.0081	0.0049	0.0068
	DRD	2.9838	0.9705	0.9397	0.9394	0.8956	0.6279	1.2504
3_100.tif	PSNR	18.08	21.44	21.57	20.96	19.18	20.17	21
	MSE	0.0156	0.0072	0.007	0.008	0.0121	0.0096	0.008
	DRD	1.47	1.0564	1.0243	0.7803	0.9546	0.7812	1.3729
4.tif	PSNR	14.28	20.78	21.1	20.14	18.29	20.47	20.28
	MSE	0.0373	0.0084	0.0078	0.0097	0.0148	0.009	0.0094
	DRD	2.9604	0.8585	0.8036	0.9884	1.0736	0.7176	1.1112

Table 3 Performance of our noise removal algorithm with different values of LT (10% noise level).

Image	Criteria	LT=2	LT=4	LT=6	LT=8	LT=10	LT=12
1.tif	PSNR	21.18	23.39	23.44	23.2	23.1	22.89
	MSE	0.0076	0.0046	0.0045	0.0048	0.0049	0.0051
	DRD	1.4754	0.6431	0.5727	0.5763	0.5826	0.5963
2.tif	PSNR	20.77	23	23.16	23.07	23.01	22.92
	MSE	0.0084	0.005	0.0048	0.0049	0.005	0.0051
	DRD	1.7087	0.765	0.6828	0.6782	0.6823	0.6891
2_100.tif	PSNR	20.48	22.26	21.94	21.55	21.28	21.03
	MSE	0.009	0.0059	0.0064	0.007	0.0074	0.0079
	DRD	1.8891	0.8331	0.7485	0.7602	0.7818	0.7991
3.tif	PSNR	20.98	23.31	23.67	23.69	23.69	23.69
	MSE	0.008	0.0047	0.0043	0.0043	0.0043	0.0043
	DRD	1.5443	0.6602	0.5529	0.5477	0.5477	0.5477
3_100.tif	PSNR	20.79	22.43	21.95	21.73	21.48	21.32
	MSE	0.0083	0.0057	0.0064	0.0067	0.0071	0.0074
	DRD	1.7344	0.7038	0.6501	0.6438	0.645	0.6522
4.tif	PSNR	21.1	21.8	21.77	21.77	21.53	21.52
	MSE	0.0078	0.0066	0.0067	0.0067	0.007	0.007
	DRD	0.8036	0.5911	0.5789	0.5698	0.5859	0.5864

6. Experimental Results and Discussion

6.1 Experiment Setup

Seven noise removal algorithms (the proposed algorithm and the six algorithms reviewed in Sect. 2) are implemented in C++ using Visual Studio 7.0. The experiment is performed on Pentium 4 PC running Windows XP.

Different real scanned images of mechanical engineering drawings are used in the experiment. A compressed file with many images used in GREC'03 contest on arc segmentation is obtained from [12]. The images contain straight lines and circular arcs with no text. The lines also have different line widths ranging from one pixel to three pixels. The four images (1.tif, 2.tif, 3.tif, 4.tif) are generated by scanning four drawings into grey level images

then thresholded using moderate value [13]. Image 2_100.tif and 3_100.tif are similar to images 2.tif and 3.tif respectively. The threshold used to convert these two images from grey to binary is small and thus creating degraded images with many thin and disconnected lines. The test images are shown in Fig. 8.

Each test image is corrupted by uniform salt-and-pepper noise at 10%, 15%, and 20% noise levels. It is worth noting that removing high levels of salt-and-pepper noise (40% for example) from binary document images is challenging due to the fact that image data (black and white) as well as the noise (salt-and-pepper) share the same small set of values (either 0 or 1) which complicates the process of detecting and removing the noise. This is different from grey images where salt-and-pepper noise could be distinguished as pixels having large difference in grey level values compared to their neighborhood. Another issue related to re-

Table 4 Performance comparison among different algorithms (15% noise level).

Image	Criteria	OpnClos	kFill	EnhkFill	ActDetec	Median	CWM 3	CWM 5
1.tif	PSNR	16.1	18.99	19.08	19.47	18.56	19.78	18.08
	MSE	0.0246	0.0126	0.0123	0.0113	0.0139	0.0105	0.0156
	DRD	2.9186	2.4453	2.3831	1.7516	1.4928	1.3808	3.1472
2.tif	PSNR	16.04	19.21	19.29	20.02	19.47	20.65	18.43
	MSE	0.0249	0.012	0.0118	0.01	0.0113	0.0086	0.0144
	DRD	3.0983	2.3694	2.3066	1.6284	1.3748	1.2663	2.9814
2_100.tif	PSNR	16.23	18.73	18.84	19.92	18.09	19.26	18.11
	MSE	0.0238	0.0134	0.0131	0.0102	0.0155	0.0119	0.0154
	DRD	2.4725	2.7315	2.6496	1.5082	1.5593	1.4494	3.4238
3.tif	PSNR	15.6	19.45	19.47	20.01	19.92	21.2	18.48
	MSE	0.0275	0.0113	0.0113	0.01	0.0102	0.0076	0.0142
	DRD	3.3259	2.2023	2.1745	1.5873	1.1987	1.0967	2.8739
3_100.tif	PSNR	17.75	19.07	19.12	20	18.76	19.53	18.38
	MSE	0.0168	0.0124	0.0122	0.01	0.0133	0.0111	0.0145
	DRD	1.687	2.5189	2.4731	1.2937	1.2346	1.2922	3.1972
4.tif	PSNR	13.96	18.13	18.17	18.12	17.31	18.9	17.32
	MSE	0.0402	0.0154	0.0152	0.0154	0.0186	0.0129	0.0185
	DRD	3.3034	1.9252	1.887	1.6552	1.4912	1.2029	2.4682

Table 5 Performance of our noise removal algorithm with different values of LT (15% noise level).

Image	Criteria	LT=2	LT=4	LT=6	LT=8	LT=10	LT=12
1.tif	PSNR	17.33	20.42	20.99	21	20.93	20.83
	MSE	0.0185	0.0091	0.008	0.0079	0.0081	0.0083
	DRD	4.0564	1.5542	1.2139	1.1239	1.1243	1.1186
2.tif	PSNR	17.59	20.37	21	21.11	21.15	21.11
	MSE	0.0174	0.0092	0.0079	0.0077	0.0077	0.0077
	DRD	3.8424	1.5984	1.2424	1.1588	1.1395	1.1424
2_100.tif	PSNR	17.35	19.9	20.17	20.03	19.84	19.67
	MSE	0.0184	0.0102	0.0096	0.0099	0.0104	0.0108
	DRD	4.4577	1.889	1.4273	1.3384	1.3301	1.3336
3.tif	PSNR	17.53	20.72	21.58	21.63	21.65	21.6
	MSE	0.0177	0.0085	0.007	0.0069	0.0068	0.0069
	DRD	3.8076	1.4374	1.0391	0.9839	0.9658	0.9716
3_100.tif	PSNR	17.56	20.24	20.33	20.25	20.1	19.97
	MSE	0.0175	0.0095	0.0093	0.0095	0.0098	0.0101
	DRD	4.3642	1.7045	1.2254	1.1343	1.1289	1.1199
4.tif	PSNR	16.75	19.11	19.52	19.64	19.61	19.55
	MSE	0.0212	0.0123	0.0112	0.0109	0.0109	0.0111
	DRD	2.9732	1.3508	1.1384	1.0615	1.0647	1.0703

moving noise from binary document images is that the number of pixels holding the image information (i.e. foreground pixels) is small compared to the total number of pixels in the image. The ratio of foreground pixels to the number of image pixels for mechanical engineering drawings is less than 10%. This means that the image information is held by less than one tenth of image pixels. If this small subset of image pixels is highly corrupted by noise, the image may not be useful (in the context of graphics recognition applications). For the test images the ratios of foreground pixels to the total images pixels are shown in Table 1.

The algorithms have parameters such as window size. To keep the size of the experiment within a manageable size, the test in this paper is limited to 3×3 window sizes for median, CWM, kFill and Enhanced kFill. The weights for CWM are selected as 3 and 5. CWM 5 is proven to re-

tain one-pixel-wide lines [6]. For Activity Detector, the default values (chosen for strong noise removal) suggested by Simard and Malvar are used [1]. Window size of 3×3 is used throughout our proposed noise removal algorithm and also for TAMd. If the information inside the window is not sufficient to decide the new value of the pixel under investigation, TAMd is to collect more information by studying the pixels adjacent to the pixel under investigation. In other words, bigger window size is less crucial under this proposed algorithm because it is taken care by TAMd. When no sufficient information is available to decide the value for a pixel p , TAMd can move the 3×3 window around the neighborhood of p to search for more clues in order to decide the proper value for it. The threshold LT will decide how far TAMd can go away from p during clue collection. In the experiment we do not determine the suitable values

Table 6 Performance comparison among different algorithms (20% noise level).

Image	Criteria	OpenClose	kFill	Enh kFill	ActDetec	Median	CWM 3	CWM 5
1.tif	PSNR	15.92	16.85	16.94	18.17	17.9	18.65	16.02
	MSE	0.0256	0.0207	0.0202	0.0152	0.0162	0.0137	0.025
	DRD	3.1649	4.2176	4.1527	2.4406	1.9242	2.0298	5.3658
2.tif	PSNR	15.49	16.89	17	18.51	18.63	19.11	16.08
	MSE	0.0283	0.0205	0.02	0.0141	0.0137	0.0123	0.0247
	DRD	3.6595	4.3683	4.3002	2.4056	1.8044	2.0371	5.4846
2_100.tif	PSNR	16.05	16.5	16.6	18.77	17.53	18.06	15.8
	MSE	0.0248	0.0224	0.0219	0.0133	0.0176	0.0156	0.0263
	DRD	2.6145	5.0697	4.9813	1.972	1.9836	2.3428	6.3393
3.tif	PSNR	15.44	16.6	16.72	18.4	18.64	19.01	15.78
	MSE	0.0286	0.0219	0.0213	0.0144	0.0137	0.0125	0.0264
	DRD	3.5245	4.523	4.4407	2.4338	1.7701	2.0969	5.7352
3_100.tif	PSNR	17.32	16.7	16.79	18.75	18.19	18.48	15.92
	MSE	0.0185	0.0214	0.0209	0.0133	0.0152	0.0142	0.0256
	DRD	1.9807	4.9107	4.8525	1.8872	1.6663	2.1177	6.2922
4.tif	PSNR	13.67	16.07	16.24	16.86	16.51	17.69	15.35
	MSE	0.0429	0.0247	0.0238	0.0206	0.0223	0.017	0.0292
	DRD	3.635	3.2869	3.2085	2.3004	1.882	1.7358	4.094

Table 7 Performance of our noise removal algorithm with different values of LT (20% noise level).

Image	Criteria	LT=2	LT=4	LT=6	LT=8	LT=10	LT=12
1.tif	PSNR	15.07	18.15	19.19	19.48	19.43	19.39
	MSE	0.0311	0.0153	0.0121	0.0113	0.0114	0.0115
	DRD	7.0799	2.9274	2.0017	1.6998	1.6409	1.6267
2.tif	PSNR	15.02	18.21	19.28	19.66	19.71	19.71
	MSE	0.0314	0.0151	0.0118	0.0108	0.0107	0.0107
	DRD	7.3775	2.995	2.0643	1.7694	1.7069	1.7069
2_100.tif	PSNR	14.83	17.58	18.61	18.83	18.84	18.76
	MSE	0.0329	0.0174	0.0138	0.0131	0.0131	0.0133
	DRD	8.3961	3.7218	2.4306	2.0527	1.9497	1.9387
3.tif	PSNR	14.83	17.78	19.04	19.45	19.57	19.6
	MSE	0.0329	0.0167	0.0125	0.0114	0.011	0.011
	DRD	7.4202	3.2646	2.1354	1.8456	1.7615	1.7425
3_100.tif	PSNR	15.03	17.8	18.53	18.79	18.74	18.68
	MSE	0.0314	0.0166	0.014	0.0132	0.0134	0.0135
	DRD	8.2955	3.5702	2.478	2.05	1.9596	1.9015
4.tif	PSNR	14.6	17.13	17.96	18.23	18.28	18.23
	MSE	0.0347	0.0193	0.016	0.015	0.0149	0.015
	DRD	5.1704	2.3928	1.7582	1.5589	1.525	1.5332

of LT directly. Instead we perform empirical experiment by varying LT values between 2 and 12. The size of the window used to calculate DRD is taken to be 5×5 .

6.2 Experimental Results

The results of our experiment are presented in the context of PSNR, MSE, and DRDM measures. Figures 9, 10, and 11 show the performance (in PSNR) of our algorithm compared to other studied algorithms on 10%, 15%, and 20% noise levels, respectively. The figures also show the performance of our algorithm with different values of LT ranging from 2 to 12. Figures 12, 13, and 14 show the performance (in DRDM) of our algorithm compared to other studied algorithms on 10%, 15%, and 20% noise levels, respectively. The figures also shows the performance of our algorithm with different values of LT ranging from 2 to 12.

Tables 2, 4, and 6 show the performance (in PSNR, MSE, and DRD) of the studied algorithms for the six test images with 10%, 15%, and 20% noise levels, respectively. Tables 3, 5, and 7 show the performance (in PSNR, MSE, and DRD) of the proposed algorithm for the six test images with 10%, 15%, and 20% noise levels, respectively. The tables also show the performance of our algorithm with different values of LT ranging from 2 to 12.

Figure 15 shows the result of applying different noise removal algorithms on one of the test images namely 1.tif. Only partial parts of the images are shown. The image is corrupted by 15% salt-and-pepper noise before applying the different noise removal algorithms.

6.3 Discussion

Getting clean images (hence high values of PSNR) when re-

Fig. 15 Image 1.tif cleaned by different algorithms.

moving noise from binary documents involves two factors i.e. removing the actual noise and retaining image detail. The former can be further expressed in two terms i.e. removing noise in the foreground/background and removing the noise that is attached to the graphical elements. Noise removal algorithms have different ability in balancing between the two factor. The two factors should be taken into consideration when designing a noise removal algorithm as well as when discussing the results of different noise removal algo-

rithms. The discussion on the performance of our algorithm compared to other algorithms is presented next.

Regarding the stability of our algorithm, it is shown in Fig. 9-a, b, d, f that high values of PSNR are observed when LT is set between 6 and 10; and it is consistent with the four real scanned test images (1.tif, 2.tif, 3.tif, 4.tif). Our algorithm performs better than other algorithms with higher levels of noise (See Figs. 9-a, b, d, f; 10-a, b, d, f; and 11-a, b, d, f). Hence our algorithm is stable.

Fig. 16 Part of image 1.tif cleaned by many algorithms and thinned by PTA2T algorithm.

Now let's consider the case of the four images corrupted by the three noise levels in more detail. From Figs. 9, 10, and 11, we can note the following trends: (i) Our algorithm performs better than the other algorithms in all images with the three noise levels. It removes the noise attached to the graphical elements while retaining thin lines. It also removes most of background noise (See Fig. 15-b). (ii) CWM 3 shows good performance (always the second highest performer) with high levels of noise (15% and

20%), but it shows oscillated performance in case of low level of noise (10%). The reason for this behavior of CWM 3 with low noise level is that the algorithm is unable to retain thin lines in the image which causes a drop in PSNR value. (iii) The performance of CWM 5 drops with increasing amount of noise. This is because the big weight for the center (considering 3*3 window) reduces the algorithm ability to remove noise compared to CWM 3. However, CWM 5 retains thin lines in the image (See Fig. 15-h).

(iv) median filter performs better than other filters with increasing amount of noise. It is known that the median filter is effective in removing salt-and-pepper noise, but it also has difficulty in preserving thin lines (See Fig. 15-f). (v) The algorithms: kFill and Enhanced kFill show similar performance where Enhanced kFill keeps an edge over kFill. They also perform moderately compared to other algorithms while their performances drops with increasing amount of noise. (vi) Activity Detector shows good ability in removing separate noise speckles but its performance is moderate compared to the other algorithms due to its inability in removing noise attached to the contour of graphical elements (See Fig. 15-e). (vii) Opening-Closing shows the lowest performance compared to all the other algorithms with all three noise levels. It removes most thin lines and it also breaks thick lines into shorter lines (See Fig. 15-i).

For the two images 1_100.tif and 3_100.tif, our algorithm still performs better than the other algorithms with 10% and 15% noise levels (See Figs. 9-c, e; and 10-c, e). Our method performs slightly better compared to the other algorithms in the case of 20% noise level (See Fig. 11-c, e). In such cases moderate value of LT is suggested.

The DRD is computed between the original clean image and the filtered image. Low value of DRD indicates less distortion, hence visually better-quality images. It is shown from the DRD values of Figs. 12, 13, and 14 that our method produces better quality images compared to other methods for most images (the four scanned images and the two degraded ones) corrupted by the three noise levels. Low values of DRD occur when LT is more than 6, which is also consistent with all test images. With 10% and 15% noise level, our method shows better image quality in the case of images 2_100.tif and 3_100.tif with higher values of LT . However, our algorithm shows lower performance compared to other methods in one case (image 3_100.tif with 20% noise level).

It can be seen from Fig. 15 that the two methods that retain thin lines while removing most of the noise are the proposed method and Activity Detector method as shown in Fig. 15-b and Fig. 15-e, respectively. The output of the Activity Detector produces cleaner background but noisy contour while our proposed method produces smother contour with little noise in the background. However, the overall performance of our algorithm in PSNR is better than Activity Detector (See Fig. 10-a). Figure 15-e contains more noise compared to Fig. 15-b. Most of the noise is attached to the contour of the graphical elements and it is visible if careful comparison between Fig. 15-a and Fig. 15-e is carried out. PSNR is sensitive to change of pixels value (even if such change is small and difficult to be observed by human visual system), hence Fig. 15-e gets lower value of PSNR compared to the proposed noise removal method shown in Fig. 15-b.

Since this work is oriented to serve our research in graphics recognition applications, we also show in Fig. 16 that the image produced by our noise removal method gives clean skeleton image after performing thinning operation [14] compared to other methods. It is worth noting

that in our previous work [15] we have proven that cleaning the images using the proposed noise removal method (called Algorithm B in the work) before performing thinning-based raster to vector conversion using commercial vectorization application gives better quality of vector data compared to other methods.

7. Conclusion

Different noise removal algorithms for binary images have been studied in this paper. A procedure (called TAMD) is proposed to enhance noise-removal operation by removing more noise and/or retaining weak features. The procedure works by tracking and analyzing a small number of pixels around end points and then deciding whether to retain or remove the tracked pixels. It gives noise removal algorithms the ability to view the surroundings of a pixel without increasing the size of the 3×3 window. This procedure is incorporated in one studied algorithm, namely, Enhanced kFill resulting in an algorithm with a new feature for dealing with thin graphical elements. TAMD could also be incorporated in noise removal algorithms' logic or used as a post-processing operation. An extensive tests on many real scanned images corrupted by three noise levels (10%, 15%, 20%) of impulsive noise are carried out. The tests show that the proposed algorithm gives better-quality images compared to other algorithms. Many figures and tables showing the performance of our algorithm are included in the paper. Distortion measurements including PSNR, MSE, and DRDM are used as the performance measurement in this experiment.

Acknowledgements

This work is supported by a Science Fund grant from the Ministry of Science, Technology, and Innovation (MOSTI), Malaysia under project number 01-01-05-SF0147. The first author is also supported by USM Fellowship.

References

- [1] P.Y. Simard and H.S. Malvar, "An efficient binary image activity detector based on connected components," Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, pp.229–232, 2004.
- [2] I.T. Phillips and A.K. Chhabra, "Empirical performance evaluation of graphics recognition systems," IEEE Trans. Pattern Anal. Mach. Intell., vol.21, no.9, pp.849–870, 1999.
- [3] A.K. Chhabra and I.T. Phillips, "Performance evaluation of line drawing recognition systems," Proc. 15th International Conference on Pattern Recognition, pp.864–869, Barcelona, 2000.
- [4] K. Chinnsarn, Y. Rangsaneri, and P. Thitimajshima, "Removing salt-and-pepper noise in text/graphics images," 1998 IEEE Asia-Pacific Conference on Circuits and Systems, pp.459–462, Chiangmai, 1998.
- [5] R.C. Gonzalez and R.E. Woods, Digital Image Processing, second ed., Prentice Hall, New Jersey, 2002.
- [6] S.J. Ko and Y.H. Lee, "Center weighted median filters and their applications to image enhancement," IEEE Trans. Circuits Syst., vol.38, no.9, pp.984–993, 1991.

- [7] L. O'Gorman, "Image and document processing techniques for the rightpages electronic library system," Proc. 11th IAPR International Conference on Pattern Recognition. Conference B: Pattern Recognition Methodology and Systems, pp.260–263, The Hague, 1992.
- [8] G.A. Story, L. O'Gorman, D. Fox, L.L. Schaper, and H.V. Jagadish, "The rightpages image-based electronic library for alerting and browsing," Computer, vol.25, no.9, pp.17–26, 1992.
- [9] L. Di Stefano and A. Bulgarelli, "A simple and efficient connected components labeling algorithm," Proc. International Conference on Image Analysis and Processing, pp.322–327, Venice, 1999.
- [10] L. Haiping, W. Jian, A.C. Kot, and Y.Q. Shi, "An objective distortion measure for binary document images based on human visual perception," Proc. 16th International Conference on Pattern Recognition, vol.4, pp.239–242, 2002.
- [11] L. Haiping, A.C. Kot, and Y.Q. Shi, "Distance-reciprocal distortion measure for binary document images," IEEE Signal Process. Lett., vol.11, no.2, pp.228–231, 2004.
- [12] L. Wenyin, Real scanned images used in Arc Segmentation Contest at GREC'03, 2006. <http://www.cs.cityu.edu.hk/~liuwy/ArcContest/test-images-2003.zip>
- [13] W. Liu, "Report of the arc segmentation contest," Graphics Recognition: Lecture Notes in Computer Science: Recent Advances and Perspectives, vol.3088, pp.363–366, Springer, 2004.
- [14] Y.Y. Zhang and P.S.P. Wang, "A parallel thinning algorithm with two-subiteration that generates one-pixel-wide skeletons," Proc. ICPR '96, vol.4, pp.457–461, 1996.
- [15] H.S.M. Al-Khaffaf, A.Z. Talib, and R. Abdul Salam, "A study on the effects of noise level, cleaning method, and vectorization software on the quality of vector data," in Graphics Recognition: Lecture Notes in Computer Science: Recent Advances and New Opportunities, eds. W. Liu, J. Llads, and J.-M. Ogier, vol.5046, pp.299–309, Springer Berlin/Heidelberg, 2008.

Hasan S. M. Al-Khaffaf gets his BSc. and MSc. in computer sciences from University of Mosul, Iraq in 1997 and 2000 respectively. Currently he is a PhD candidate in School of Computer Sciences/Universiti Sains Malaysia working on the field of graphics recognition. He is a Provisional Member of Australian Computer Society (ACS). His interests include analysis of engineering drawings, raster to vector conversion process and its performance evaluation.

Abdullah Z. Talib obtained B. Sc. (Hons.) in Mathematical Sciences from the University of Bradford, Britain in 1983, M. Sc. in Computing Science from the University of Newcastle upon Tyne, Britain in 1985 and Ph.D. in Computer Science from the University of Wales in Swansea, Britain in 1995. He started his career as a university lecturer at the University of Science Malaysia (USM) in 1986 and promoted to the current position as an Associate Professor in 2003. He has also served as chairperson for

computer science, information systems and computing science program at the School of Computer Sciences, USM. Currently he is the deputy dean for academic and student development at the same school. His research interests include graphics and visualization, geometric computing, computational modeling and intelligent systems. He has published over 50 papers in conferences and journals, and served as conference organizing chair, member of program committees for many international/national conferences. He has also reviewed several conference and journal papers.

Rosalina Abdul Salam is associate professor at the School of Computer Sciences, Universiti Sains Malaysia and a member of Artificial Intelligence Research Group. She received her Bachelors degree in Computer Science in 1992 from Leeds Metropolitan University, United Kingdom. She was a system analyst in Intel Penang, from 1992 to 1995. She returned to United Kingdom to further her studies. She received her Masters degree in Software Engineering from Sheffield University, United

Kingdom in 1997. She completed her Ph.D. in 2001 from Hull University in the area of computer vision. She has published more than 60 papers in journals and conferences. She is a member of International Computational Intelligence Society and World Enformatika Society. Recently she joined the editorial board of the International Journal of Computational Intelligence and the International Journal of Signal Processing. Presently, she is continuing her teaching, graduate supervisions and her research. Her current research area is in the area of artificial intelligence, image processing and bioinformatics applications. The most recent project that she is working is on underwater images and cellular images.