## *A. Pre-processing* (Constrat Estimation)

Based on the gray-level morphological operators in Section 2.2, the morphological closing function is used to extract the contrast intensity. To ensure that the contrast image contains all pixels that lie within the text area, the pixel intensity values in the document image are adjusted. Simply, all gray-levels are mapped to a new range from 0 to 255. Fig. 9 (a) shows an unevenly illuminated, bleeding-through document image and its adjusted one is presented in Fig. 9 (b). Fig. 9 (c) illustrates the background image extracted by the gray-level morphological closing operation. By subtracting the original image from the background one, the contrast image is generated, as in Fig. 9 (d).

## *B. Double Threshold Binarization*

**Algorithm1:** ***1-D histogram Shannon entropy-based thresholding***

**Input**: A grayscale image

**Output**: A threshold value $T$

**Procedure**:

MAXIMUM=0; $T$=0;

for each $t$=0, $t$<$L$, $t$++

$$P(t)=\sum_{i=0}^{t} p_i;$$

$$H(S)=H_b+H_w=-\sum_{i=0}^{t}\frac{p_i}{P(t)}\log\left(\frac{p_i}{P(t)}\right)-\sum_{i=t+1}^{L-1}\frac{p_i}{1-P(t)}\log\left(\frac{p_i}{1-P(t)}\right);$$

if $H(S)$> MAXIMUM then

MAXIMUM=$H(S)$;$T$=$t$; end if \ end for

**Algorithm 2** ***Ternarization system using 1-D histogram Shannon based entropy***

**Input**: A grayscale image
**Output**: Two threshold values, $T_1$, and $T_2$
**Procedure**:
MAXIMUM_K = 0; $T_1 = 0$; $T_2 = 0$;
for each $t_1 = 0$, $t_1 < L\text{-}1$, $t_1$ ++
for each $t_2 = t_1+1$, $t_2 \le L\text{-}1$, $t_2$ ++

$$P_b = \sum_{i=0}^{t_1} p_i \;;\; P_{bw} = \sum_{i=t_1+1}^{t_2} p_i \;;\; P_w = \sum_{i=t_2+1}^{L-1} p_i$$

$$H(S) = -\sum_{i=0}^{t_1} \frac{p_i}{P_b} \log(\frac{p_i}{P_b}) - \sum_{i=t_1+1}^{t_2} \frac{p_i}{P_{bw}} \log(\frac{p_i}{P_{bw}}) - \sum_{i=t_2+1}^{L-1} \frac{p_i}{P_w} \log(\frac{p_i}{P_w});$$

if $H(S)$ > MAXIMUM_K then
MAXIMUM_K = $H(S)$; $T_1 = t_1$; $T_2 = t_2$;
end if \ end for\ end for

**function** ***Relabel_Img***

Input: the original image $I$, the contrast image $C$
Output: Relabel the near-text pixel in the image $I$
for $i = 1 : M$
for $j = 1 : N$
if $(C(i,j) \geq T_1)$
$[Num, I_{Mean}, I_{Std}] =$ ***Relabel_Reg***$(I, C, i, j)$;
if ( $I(i,j) <$ min $(I_{Mean} + I_{Std}, T_2)$ and $(Num > 0)$ then
$I(i,j) = 1$;
else
$I(i,j) = 0$;
end if
$i = i + w$;
$j = j + w$
end if \ end for \ end for

**function** $[Num, I_{Mean}, I_{Std}] = Relabel\_Reg(C, i, j)$

Consider the $(2w+1) \times (2w+1)$ window of the contrast image $C_w = C(i - w: i + w, j - w: j + w)$
Consider the $(2w+1) \times (2w+1)$ window of the original image $I_w = I(i - w: i + w, j - w: j + w)$
$Num = 0$; $I_{Mean} = 0$; $I_{Std} = 0$;
for $x = 1 : 2w+1$
  for $y = 1 : 2w+1$
    if $(C_w(x, y) \geq T1)$ then
      $C_w(x, y) = 1$;
      $Num$ ++;
    else
      $C_w(x, y) = 0$;
    end if \ end for \ end for
  if $(Num > 0)$ then

$$I_{Mean} = \frac{I_w \times C_w}{Num} \text{ and } I_{Std} = \sqrt{\frac{C_w \times (I_w - I_{Mean})^2}{Num}}$$

  end if

## *C. Post-processing*

The post-processing operations are applied on the binarized image in order to eliminate noise, fill the breaks, gaps or holes, and preserve stroke connectivity. Two kinds of noise are considered in our work. The first one is “salt and pepper” noise which is defined as (1) white regions that are smaller than the stroke width and that usually stay inside the character stroke and (2) black regions that are smaller than the stroke width and that settle around the document text or that scatter over the image. Fig. 11 (a) shows some examples of “salt and pepper” noise as defined in our scheme. Fig. 11 (b) illustrated some cases of another kind of

noise, called “block” noise, which consists of the black regions with a window size bigger than the stroke width. According to the experiments, the “block” noise was often located along the border of the images from the ICHFR 2010 database [34]. These images contained most kinds of problems that could be met in old documents: presence of stains and strains, backgrounds with big variations and uneven illumination, ink seepage, etc. To deal with the two different kinds of noise, two strategies have been applied in two phases and are described as follows:

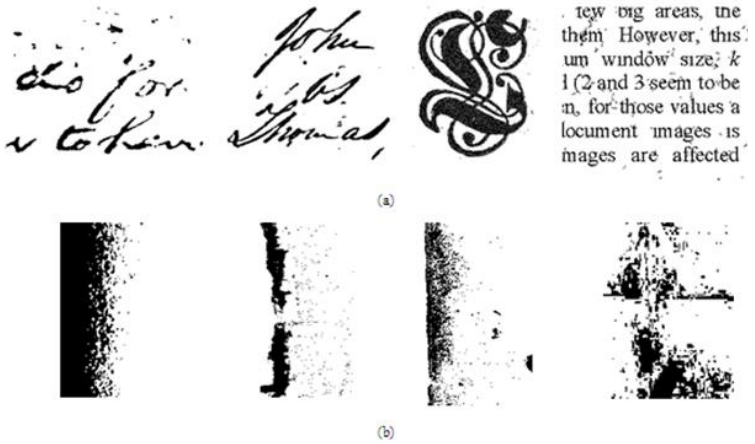


Figure 11. Representative examples of noise: (a) “salt and pepper” noise and (b) “block” noise

**Strategy 1**: To deal with the first kind of noise – "salt and pepper" noise.

Firstly, the binary morphological operators are applied to bridge the unconnected pixels and to remove the isolated or spurious pixels. Fig. 12 gives some representative examples of binary morphological operations in which the second row has the corrected version of the first row.

Secondly, we correct the areas that are larger than one pixel and smaller than the character stroke width by using shrink and swell filters. These filters can remove the region noise from the background, especially around the text, as well as fill the foreground region. Take an $R_1 \times R_2$ region as an example: Let *Num*_1 and *Num*_2 be the number of white

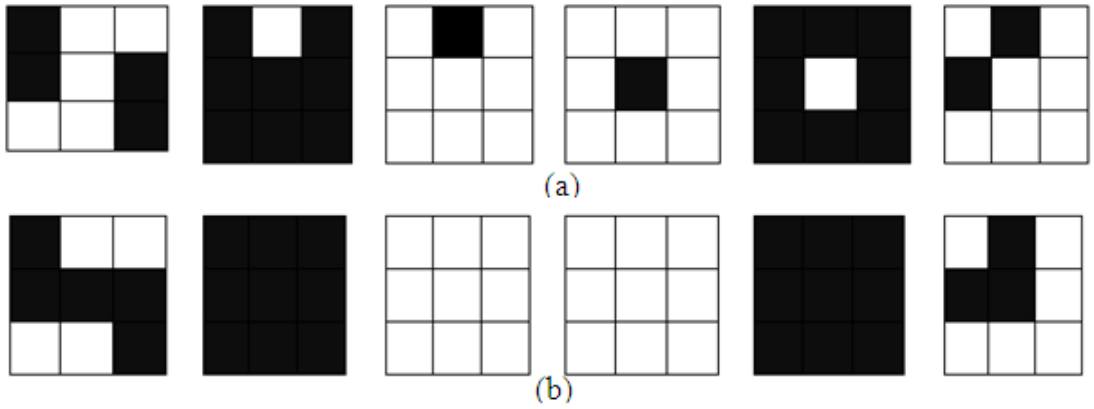


Figure 12. Examples of (a) unconnected, isolated, spurious pixels, (b) corrected ones by the binary morphological operators

pixels inside $(R_1 \times R_2)$ and $(R_1 - 1) \times (R_2 - 1)$ windows, respectively. Simply, if *Num*_1 equals *Num*_2, then window $(R_1 - 1) \times (R_2 - 1)$ becomes black. If *Num*_1 equals [*Num*_2+2$(R_1 + R_2 - 2)$] then window $(R_1 - 1) \times (R_2 - 1)$ is filled in white.

**Strategy 2**: To deal with the second kind of noise – "block" noise.

As defined, the block noise consists of black regions that are bigger than the stroke width; therefore, a filter cannot work well because of the unwanted effects on document text areas. To deal with such kind of noise, a graph searching strategy is utilized. A graph (also called a tree) is defined as follows: (1) ***Root***: a complete black block, (2) ***Node*** (internal/external): a block in which the number of black pixels is larger than $2 \times w$, and (3) ***Edge***: two adjacent black pixels that belong to two different and adjacent nodes. Each node belongs to only one tree (graph).

The graph searching strategy is performed as in **Algorithm 4**

**Algorithm 4***: The graph searching strategy*

**Input**: A binarized image

**Output**: A set of trees. Each tree is considered to be a block noise.

**Procedure**:

- The binarized image (as in Section 4.3) is divided into non-overlapping blocks with the window size bigger than the character stroke width $w$.
- Scan the image from left to right, top to bottom. For each block $X$:
  - If $X$ is completely black and $X$ does not belong to any tree, then
    - Set $X$ as root of a new tree
    - For each node inside the tree, called $A$, consider its 4-neighbouring blocks, denoted by $A_i$, (for $i = 1, 2, .., 4$). If $A_i$ is a node and there is an edge between $A$ and $A_i$ then add $A_i$ to the tree $X$.

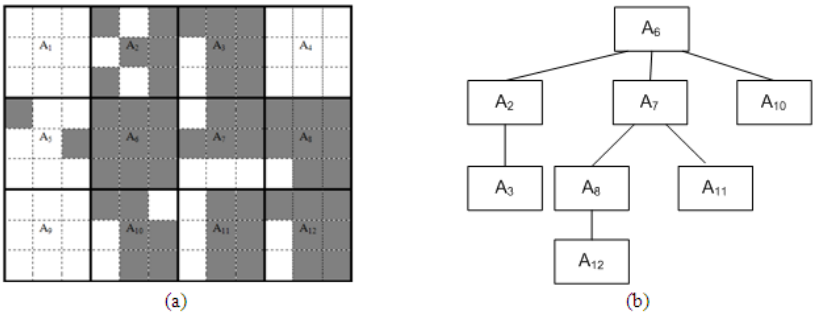


Figure 13. Examples of graph searching strategy where (a): binarized image and (b): tree

Take Fig. 13 as an example. Fig. 13 (a) presents the binarized image. The character stroke width is assumed to be $w = 2$. The image is divided into 12 non-overlapping 3x3 blocks, called $A_i$ (for $i$=1, 2, .., 12). Scan the image from left to right, top to bottom, $A_6$ is the complete black block and does not belong to any tree. Set $A_6$ as a root of a tree.

For each node of the tree = $\{A_6\}$, consider the 4-neighbouring blocks of $A_6$, $\{A_2, A_7, A_{10}\}$ are nodes and $A_5$ is not a node by definition. Thus, the tree becomes $\{A_6, A_2, A_7, A_{10}\}$.

Next, consider the 4-neighbouring blocks of $A_2$, $A_3$ is a node and $A_1$ is not a node. Thus, the tree becomes: tree = $\{A_6, A_2, A_7, A_{10}, A_3\}$.

Similarly, we fill the tree with the root $A_6$. The tree is shown in Fig. 13 (b).

After the graph searching algorithm is applied to the image, all the trees (graphs), which have been detected, are removed from the image. In order to avoid the possibility of removing text, we have used the window size bigger two times the average stroke width.