Noise Removal and Image extraction Report

24 June, 2009

Prepared by: Mr. ING, LENG IENG
Miss KHEM, SOCHENDA

Cambodia Country Component
PAN Localization project
PAN Localization Cambodia (PLC) of IDRC

Table of Contents

1. Introduction

Digital images are prone to a variety of types of noise. Noise is the result of errors in the image acquisition process like scanning or shooting which results in pixel values that do not reflect the true intensities of the real scene. There are several ways that noise can be introduced into an image, depending on how the image is created such as amplifier noise (Gaussian noise), salt and pepper noise, shot noise, film grain, etc. For example:

If the image is scanned from a photograph made on film, the film grain is a source of noise. Noise can also be the result of damage to the film, or be introduced by the scanner itself.

If the image is acquired directly in a digital format, the mechanism for gathering the data (such as a CCD detector) can introduce noise. Electronic transmission of image data can also introduce noise.

In this work, noises are considered as the unwanted block of black pixels. Those blocks are considered to be very small comparing to the size of the block of the characters.

2. Literature Review

2.1. Connected Component Labeling

Everything on earth is interconnected no matter what type of object it is. Likewise, each printed pixel on paper has also connections. Such connections make those pixels unify, and thus form what a human understands as a shape. A shape that is visible to human’s eyes can be everything that appears on paper from a tiny component such as a dot to a larger component such as an individual letter, a picture, a line, etc. In OCR processing, each connected component must be extracted.

Connected Component Labeling is one of the techniques used to uniquely label each connected component, particularly a shape. This technique scans the image pixel by pixel or runs of pixels in order to group pixels based on pixel connectivity. Two distinctive algorithms are used to label the connected component.

2.1.1. Flood fill Algorithm

Flood fill, also known as seed fill, is an algorithm that uses the given pixel to find other pixels that connect to it, and then fills them with the same color as the given pixel. Flood fill is nowadays used as a bucket fill tool in many paint programs[1]. It can be implemented in many ways such as using a recursive function, a stack, or even a more efficient in terms of stack-friendly method called a recursive scan line flood fill algorithm. Flood fill may be used to label the connected component in an image. The strength of this method is that there is no complexity in the implementation. However, the drawbacks to this are the huge consumption of memory and stack overflow issues.

There are numbers of algorithm in regard to this algorithm such as 4-way recursive method, 8-way recursive method, 4-way method with stack, 8-way method with stack, recursive scan line flood fill algorithm, and scan line flood fill algorithm with stack[2].

2.1.2. Two-pass Algorithm

The algorithms discussed can be generalized to arbitrary dimensions, albeit with increased time and space complexity. Relatively simple to implement and understand, the two-pass algorithm iterates through 2-dimensional, binary data. The algorithm makes two passes over the image: one pass to record equivalences and assign temporary labels and the second to replace each temporary label by the label of its equivalence class.

On the first pass:

1. Iterate through each element of the data by column, then by row
2. If the element is not the background
3. Get the neighboring elements of the current element
4. If there are no neighbors, uniquely label the current element and continue
5. Otherwise, find the neighbor with the smallest label and assign it to the current element
6. Store the equivalence between neighboring labels

[1] Flood fill, from Wikipedia, the free encyclopedia: http://en.wikipedia.org/wiki/Flood_fill
[2] Lode's Computer Graphics Tutorial: Flood Fill, http://www.student.kuleuven.be/~m0216922/CG/floodfill.html#Introduction_

On the second pass:

1. Iterate through each element of the data by column, then by row
2. If the element is not the background
3. Relabel the element with the lowest equivalent label

Here, the **background** is a classification, specific to the data, used to distinguish salient elements from the **foreground**. If the background variable is omitted, then the two-pass algorithm will treat the background as another region.

2.2. Existing Noise Removal Methods

Noise removal is necessary for any process, i.e. speech processing, image processing. It is the important step in any system. In image processing, scanned documents always ported with noise from the scanner or the documents themselves. Different methods are better for different kinds of noise. The methods available for reducing noise in image are:

- Linear Filtering
- Median Filtering
- Adaptive Filtering

3. Methods Proposed

In this work, noise in binary scanned documents is considered as the unwanted blocks of pixels that are not supposed to exist in the document but introduced by the scanner. A very simple technique is proposed by firstly detect the region of every connected component in the documents. The obtained connected components whose area is smaller than the defined threshold will be replaced by the white pixels. To detect the region of every connected component, Connected Component Labeling is applied. Two methods to detect the connected components are applied in this work: Two-pass Method and Flood Fill.

Stack over flow is the risk that always caused by recursive algorithm which is implemented using the Flood Fill algorithm. To improve the performance of the program, line by line must be scanned to get the connected components. Two-pass Algorithm with Union-Find is chosen.

3.1.1. Terminologies

Before starting the explanation of the method, it is an essence to get familiar with some terminologies used in our algorithm.

1. **Node**: an alternative term for pixel.
2. **Root node**: a node which is the parent of other node. Its rank is 0, indicating the top node in the tree.
3. **Parent node**: a node which has some other nodes attach to it. Parent node is not to be confused with the Root node since its rank can be any value ranging from 0 if it is a root node to n value, where is the unsigned integer value other than 0 if it is under the root node but has other nodes as the child.
4. **Child node**: a node whose parent is the other node. Its rank starts from 1 to n, where n is the unsigned integer value other than 0.
5. **parent***:* an element of node structure type in the `ConnectedComponent` class used to point to the parent node.
6. **index***:* an element in the `ConnectedComponent` class used to store the element number. This means that if the 9 nodes are in the stored, the 10th one would have the *index* value of 10.
7. **label***:* an element in the `ConnectedComponent` class used to store the assigned label of the node.
8. **rank***:* an element in the `ConnectedComponent` class used to identify the rank of the node. If any node has *rank* equal to 0, it is a root node. Hence, the smaller *rank* indicates the higher level of the node.
9. **actual row***:* an element in the `ConnectedComponent` class used to store the actual row index of the node (pixel) in the actual image.
10. **actual column***:* an element in the `ConnectedComponent` class used to store the column actual index of the node (pixel) in the actual image.

3.1.2. Two-pass Method Algorithm

Two-pass method, as the name suggests, is an algorithm which scans through the whole image twice from top to bottom, left to right. The first pass involves the process of assigning the label to each foreground (black) pixel in accordance with the following conditions:

1. If it is a background (white) pixel, create a new node for the pixel but the node is reserved for nothing other than used to fulfill the pixel index in the image. Thus, this node would contain these elements: parent = null, index = new (incremental) index value, label = 0, rank = 0, actual row = 0 and actual column = 0. In our code, AddElement() is an applicable method for this condition. This method belongs to ConnectedComponent class.
2. If it is not a background, the pixel is black; then do the following:
   - If neighbors are not labeled, assign a new label to the current pixel.
   - If at least one of the neighbors has label, find the smallest label and copy it to the current pixel. Then, mark the two as equivalent by using the union-find method.

The following figure illustrates what neighbors to be checked. ***P*** is the current pixel and ***x*** is neighbor to be checked.

x	x	x
x	**P**	

Figure 1 Four neighbors (x) to be checked for the current pixel (P)

3.1.3. Union-Find Method Algorithm

Union-Find method is a dual-purpose method used for both finding the root node of the current node and combining two trees.

3.1.3.1. Union

Union algorithm is the process of merging two trees that have the same root node. The merging process is done by always connect the smaller tree to the larger tree, as described in Connected component labeling – Part 6, Steve on Image Processing. This shortens the paths in the trees.

```
void ConnectedComponent::Union(int setIndex1, int
setIndex2)
{
	if(setIndex1 != setIndex2)
	{
		Node^ set1 = nodes[setIndex1];
		Node^ set2 = nodes[setIndex2];
		if(set1->rank > set2->rank)
		{
			set2->parent = set1;
		}
		else if(set1->rank < set2->rank)
		{
			set1->parent = set2;
		}
		else
		{
			set2->parent = set1;
			set1->rank++;
		}

		numOfSets--;
	}
}
```

3.1.3.2. *Find*

Find algorithm is the process of walking from the given node to its root node. The function returns the *index* (see 3.2.1 Terminologies) value of the root node.

```
int ConnectedComponent::FindSet(int elementIndex)
{
	Node^ curNode = nodes[elementIndex];

	// Finds the root of the set to which the element
	// belongs
	while(curNode->parent != nullptr)
		curNode = curNode->parent;
	Node^ root = curNode;

	curNode = nodes[elementIndex];
	while(curNode != root)
	{
		Node^ next = curNode->parent;
		curNode->parent = root;
		curNode = next;
	}

	// Release memory
	delete curNode;

	return root->index;
}
```

4. Experimental Results

The experiments were performed on a set of scanned local news paper. The result shows that the implemented method can be used to remove the images and noise produced by the scanner which are well separated from the text. The texts which are connected to the images due to black noise are considered as the region to be removed.

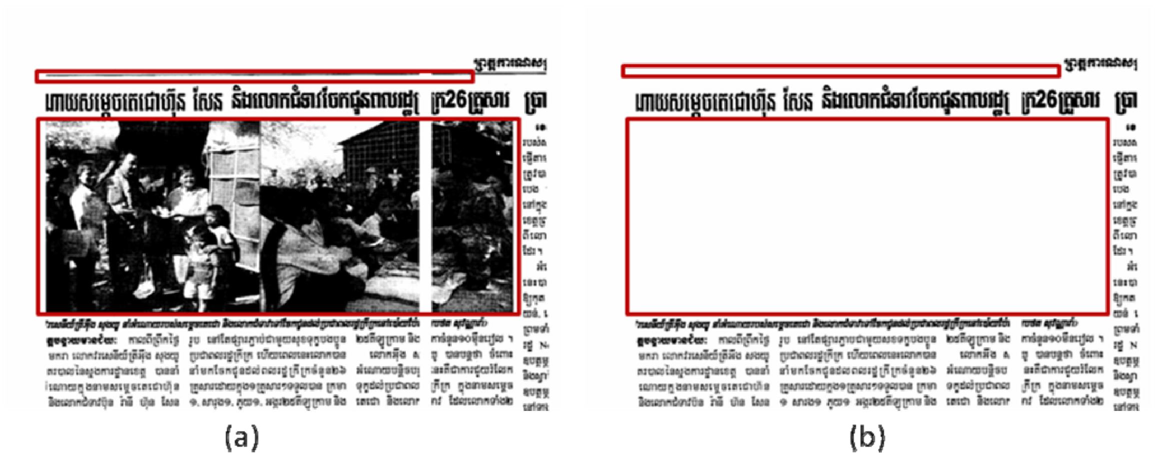
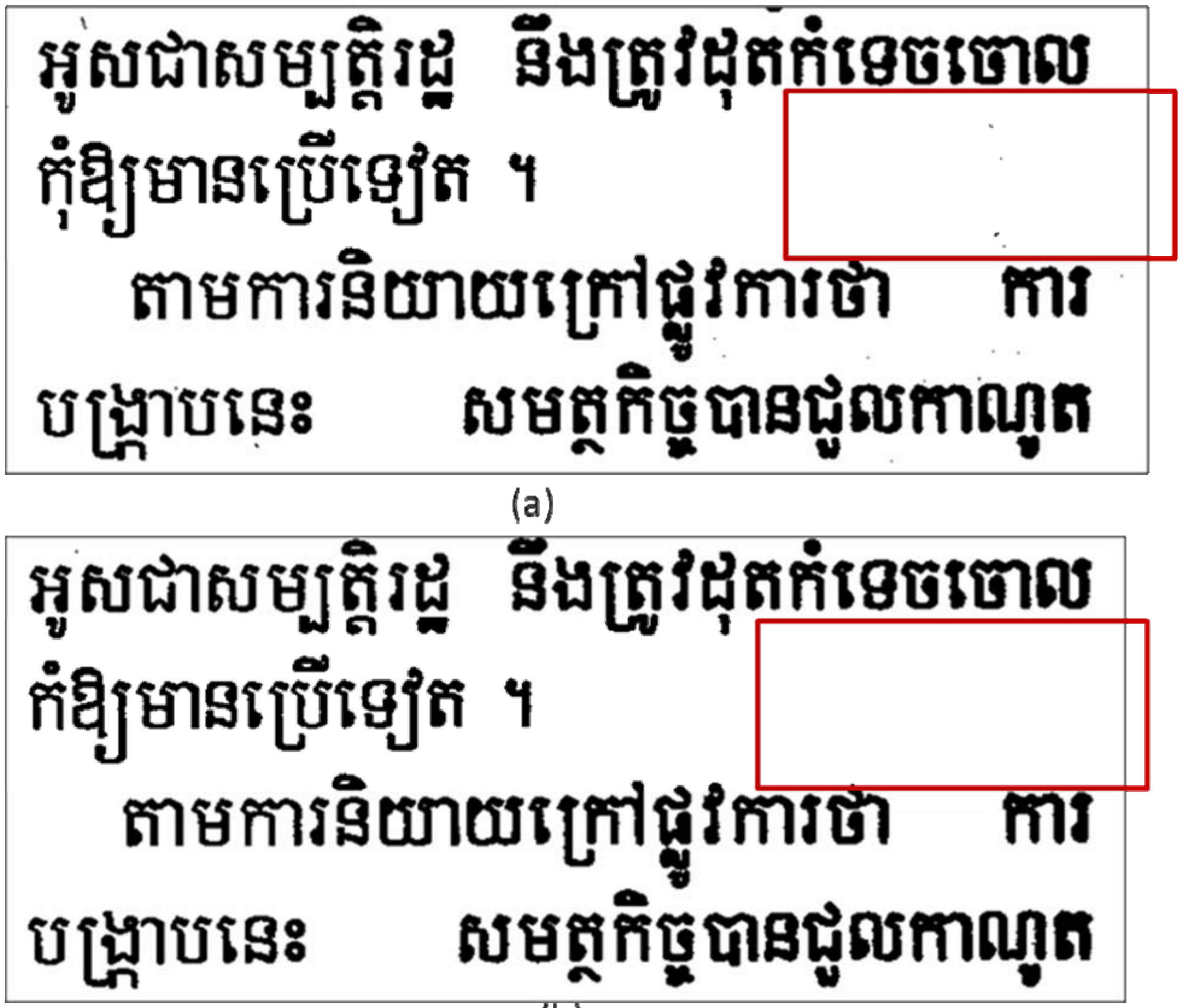


Figure 2: (a) input document with image, and (b) the output document after removing the image.

Figure 2.(a) shows the input document contained some images and lines, and Fig.(b) shows the output document after removing the lines and images. It can be seen that the images can be eliminated from the documents by determining the threshold of the images' sizes.



អូសជាសម្បត្តិរដ្ឋ នឹងត្រូវដុតកំទេចចោល
កុំឱ្យមានប្រើទៀត ។
តាមការនិយាយក្រៅផ្លូវការថា ការ
បង្ក្រាបនេះ សមត្ថកិច្ចបានជួលកាណូត

(a)

អូសជាសម្បត្តិរដ្ឋ នឹងត្រូវដុតកំទេចចោល
កុំឱ្យមានប្រើទៀត ។
តាមការនិយាយក្រៅផ្លូវការថា ការ
បង្ក្រាបនេះ សមត្ថកិច្ចបានជួលកាណូត

(b)

Figure 3: (a) Input document with noise produced from scanner, (b) output document after removing noise

Figure 3.(a) shows the input document contained some noise produced from scanner, and Fig.(b) shows the output document after removing the unwanted noise. It can be seen that the noise in the document can be eliminated from the documents by applying thresholding.

5. Conclusion

The noise removal algorithm has been proposed by firstly define the connected components in the images, and then replace the unwanted black pixel region which has the area smaller than the defined threshold with white ones. From the experiments, the method is acceptable to reduce the images and noise which are detected as different region by the connected component from the documents. However, the improvement on image enhancement must be taken into account in the future work because the degradation of the image can cause not only the replacement of white pixel to black pixel, but also from black to white pixel.

6. References

1. Euripides G.M., P. *Binary Image Analysis*. Retrieved June 16, 2009, from Technical University of Crete: Computer Vision web site: www.intelligence.tuc.gr/~petrakis/courses/computervision/binary.pdf
2. LengIeng, I., Sochenda K. (2009). Noise Removal Using Connected Component Labeling. PAN Localization Cambodia of IDRC. Phnom Penh, Cambodia.
3. Steve, E. (2007). Connected component labeling – Part 5: Two-pass method. Retrieved June 17, 2009, from Matlab Central: Steve on Image Processing web site: http://blogs.mathworks.com/steve/2007/05/11/connected-component-labeling-part-5/
4. Steve, E. (2007). Connected component labeling – Part 6: Union-Find. Retrieved June 17, 2009, from Matlab Central: Steve on Image Processing web site: http://blogs.mathworks.com/steve/2007/05/25/connected-component-labeling-part-6/
5. Union-Find Algorithms. Retrieved August 31, 2009, from Princeton University: Department of Computer Science web site: http://www.cs.princeton.edu/courses/archive/spr09/cos226/lectures/01UnionFind.pdf