AN EFFICIENT BINARY IMAGE ACTIVITY DETECTOR BASED ON CONNECTED COMPONENTS

Patrice Y. Simard and Henrique S. Malvar

Microsoft Research
One Microsoft Way, Redmond, WA, 98052, USA

ABSTRACT

Activity detection on binary images can be a useful part of image processing for detecting noise, texture, printed text, or dithering. In this paper we present an image activity detector based on computing a density of selected connected components (CCs). Connectedness is a useful property because it is present in individual printed letters, lines, and edges. In contrast, salt-and-pepper noise and dithering are typically composed of a large number of disconnected patterns. By filtering the CCs based on size, we can measure different kind of activities, and segment or filter the image accordingly. The activity detector is extremely efficient and can be run in a fraction of the time it takes to compute a run-length encoding version of the image. As an example, we built a noise removal filter based on the density of CCs, which is both faster and better than a conventional median filter.

1. INTRODUCTION

Electronic documents are often formatted as binary images, which commonly contain some degradation. Dithering is used to convert color and gray level information to binary. Bleeding and erosion happens when the image is converted to a lower resolution. Salt-and-pepper noise result from inappropriate scanning settings, paper granularity, pixel noise, etc.

Text and line art are connected, because it makes them easier to draw and identify by humans. Thus, connectedness is a feature of human-produced visual information. Large connected component can also be produced by special dithering patterns.

In this paper we present a fast algorithm to compute an activity measure that indicates how busy a document is in a small neighborhood of each pixel. The slowest part of the algorithm is the computation of connected components (CCs), which can be done in almost linear time. We apply the activity measure to build a noise removal algorithm that is faster than even a simple median filter, but whose performance approaches those of more sophisticated systems [1]–[4].

2. ACTIVITY DETECTOR

In its simplest form, activity detection assigns to each pixel an activity level, whereas activity measures the degree of busyness around each pixel. In noise removal applications, areas of low activity can be filtered more strongly. In areas of high activity, filtering should be lighter to avoid removal of dithering patterns or small features such as dots of i's [1], [2].

If we compute activities via combination of highpass and median filters, as usual, we incur the expense of running such filters at every pixel, which is time-consuming for high-resolution documents. Complexity gets worse if we add dilations operators for smoother activity measures [2] or morphological [3] or other post-processing operators [4].

For faster processing, we can first represent each scan line by a run-length representation. If the operations to be performed can be computed on the run-length representation, significant speedups can be achieved, since there are many fewer runs than there are pixels. Another disadvantage of computing activity measure at the pixel level is that they usually cannot benefit from long strokes in line drawings or large fonts.

To reduce the complexity of computing activity detectors, we propose a two step-approach: first we determine the connected components in the bitmap, which can be done in the run-length domain. Then we compute activity measure for a pixel by counting the number of CC bounding boxes that intersect a region around that pixel. That makes the computational complexity proportional to the number of connected components, which is typically orders of magnitude lower than the number of pixels. An arguably positive side effect is that every CC contributes equally to activity, e.g. all printed letters have the same activity contribution even though a 'W' is more complex than an 'I'. Similarly, long strokes from line art increase less the activity measure than they do in pixel-based methods. That way, filtering operators can be more aggressive in the vicinity of such strokes.

We compute CCs from a run-length representation of the binary image. The CCs are computed in O(n log*(n)) where n is the number of runs, and log* is the inverse of the Ackermann function, using the union-find algorithm [5]. This algorithm is for all practical purposes linear in the number of runs. Next, we compute the bounding boxes of all CCs; this step is illustrated in the top line of Figure 1. Then we compute, for each square neighborhood of radius L around each pixel, how many CCs bounding boxes intersect that neighborhood. This step would be expensive if the computation was done for each pixel independently. Instead we compute the contribution of each CC to an

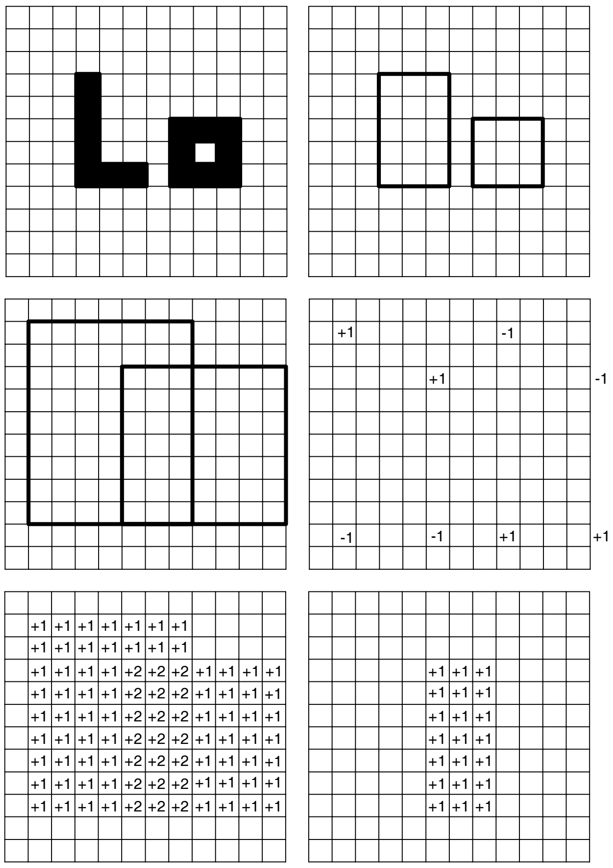


Figure 1. Top: original image (left) and CC bounding box (right); middle: bounding box expanded by 2 (left), and derivative contribution (right); bottom: integral (left) and threshold activity (right).

activity map. For instance if we want to compute how many CCs intersect a neighborhood of radius 2 around a pixel (a 5 by 5 neighborhood), we expand the bounding box of each CC by 2 (Figure 1, middle left), and add 1 to each pixel in the activity map corresponding to the enlarged region (Figure 1, bottom, left). However, this computation can be further optimized by computing the X and Y derivative of each CC contribution and adding the derivative to the derivative of the activity map (Figure 1, middle right). This latter operation takes only 4 additions per CC and is therefore very efficient. Once all the CCs have contributed to the derivative of the activity map, the derivative is integrated with respect to X and Y, to yield to true activity map (Figure 1, bottom, left). The activity map can then be thresholded for segmentation (Figure 1, bottom right).

Further speed up can be obtained by subsampling the activity map. Since the activity map is typically thresholded for segmentation, we often do not need high pixel accuracy. With little loss in precision, one can therefore divide the position of each contribution of each bounding box by 2 or 4, and compute the activity map integral on an image of 1/4 or 1/16 of the original size and achieve the corresponding speedup.

To compute diverse activity maps, we can filter the CCs according to size. For instance, in one activity map, we can sum the contributions of CCs of size less then R_1 pixels, while in another map we can sum the contributions of CCs of size less than R_2 pixels. Each activity map can then be thresholded differently, as discussed in the next section.

Computing the run-length encoding is proportional to the number of pixels, but is very fast, since it takes only a test and a counter update. Computing the CCs is proportional to the number of runs (about 15 operations per run). Depending on the number of CCs, this operation is typically comparable in time to computing the run-length encoding. Filtering the CCs and adding their contribution takes negligible time because it is only a few operations per CCs. Computing the activity map is a double integral (2 additions per pixel), which is typically done at half resolution (one 4th of the operations) or 1/4 resolution (one 16th of the operations).

3. APPLICATION TO SALT-AND-PEPPER NOISE REMOVAL

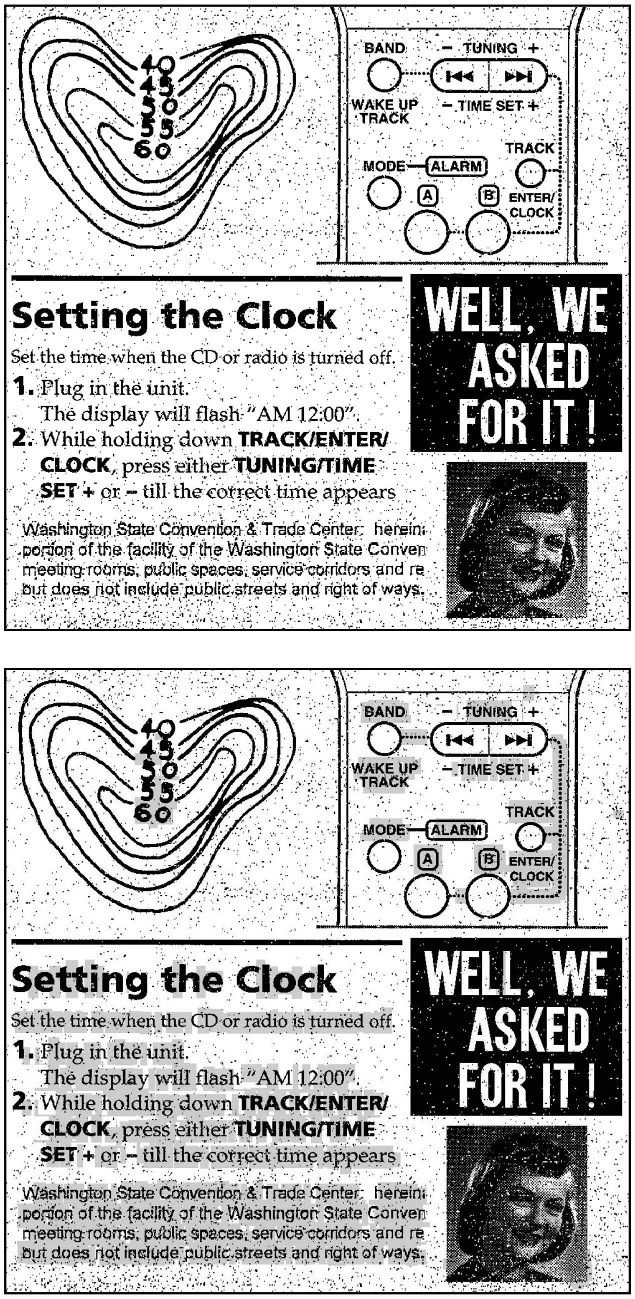
We applied the algorithm described in Section 2 to the noisy 1,000×1,000-pixel image in Fig. 2, which was assembled from a collection of scans from the same noisy scanner. In the bottom figure we highlight in gray the pixels for which the text activity measure is greater than 2. That process would be a front end for text segmentation, but in itself does a reasonable job in segmenting text. Note that the line drawings do not generate regions of high activity, whereas many activity detectors reported to date include graphics and text in the same category [1], [2].

A simple salt-and-pepper noise removal system can be build from the algorithm in Section two, in the following way:

1. Determine D_1 as the set of CCs for which the activity measure is greater than threshold A_1, but only CCs with more than B pixels are included in the measure. That set is likely to include pixels near text areas or dithered areas.
2. Determine D_2 as the set of CCs for which the activity measure is greater than threshold A_2. If A_2 is set high enough, that set is likely to include only pixels in dithered areas.
3. Remove CCs with fewer than R_1 pixels, except those in D_1 or D_2 and also CCs with fewer than R_2 pixels, except those in D_2.

The third step basically states that in background areas which may contain lines (but not text or dithering) we can remove larger CCs than in the areas containing text or dithering. Also, in regions of dithering we do not remove any CCs.

We compared our noise remover above with a simple system based on median filtering:


40
45
50
55
60
BAND
– TUNING +
WAKE UP TRACK
– TIME SET +
TRACK
MODE
ALARM
A
B
ENTER/ CLOCK
WELL, WE ASKED FOR IT !

Setting the Clock

Set the time when the CD or radio is turned off.

1. Plug in the unit.
The display will flash "AM 12:00".

2. While holding down **TRACK/ENTER/ CLOCK**, press either **TUNING/TIME SET +** or – till the correct time appears

Washington State Convention & Trade Center: herein portion of the facility of the Washington State Conven meeting rooms, public spaces, service corridors and re but does not include public streets and right of ways.

Figure 2. Top: original noisy image used for the experiments; bottom: image where pixels of high text activity are highlighted in gray.

1. From the input image x, compute a difference image u by double binary differencing (i.e. x-or from previous pixel, across rows and columns).
2. Given u, compute an activity image a by adding the number of u pixels equal to 1 in a neighborhood of dimensions $2L_2+1 \times 2L_2+1$.
3. Compute the final filtered image y by median filtering the input image x, i.e. flipping the values of pixels that are different from the median computed in a region of dimensions $2L_1+1 \times 2L_1+1$, but only in regions where the activity is below a threshold A_0.

This noise remover above is one of the simplest in terms of computational complexity. Better results can be obtained with more sophisticated system, but at a high computational cost [1]–[5].

In Fig. 3 we show the results of processing the noisy image of Fig. 2 with the median filter and the CC-based filter defined above, for the following choice of parameters (for a strong noise reduction): L_1=1, L_2=10, A_0=85; A_1=2, A_2=30, R_1=20, R_2=5. We see that in graphics and flat regions the CC-based result is much cleaner. In the text areas, the CC-based system has also a cleaner output, without blurring the characters as in the median filter. For the dithered regions, the two systems have comparable performance. For the median filter, 2.6% of the pixels were modified, whereas for the CC-based system only 1.8% of the pixels were modified. So, even though the CC-based system modified ~30% fewer pixels, it produced a much cleaner output; in other words, it made better decisions about which pixels to be modified.

4. CONCLUSION

We have presented an activity detector which is based on counting connected components rather than pixel activity. This detector has good segmentation properties, while being more efficient than pixel-domain filters. We tested the segmentation by building a salt-and-pepper noise remover based on connected component activity. When compared to a pixel-based noise remover of similar complexity, it yields better results with a faster running time.

One of the interesting properties of our activity detector is that once the connected components have been extracted, we can compute multiple activity maps very efficiently. We believe that accurate dithering detection across gradients, multi-scale text detection, or complex texture segmentation, cannot be performed with a single activity map. Our approach allows easily parametrizable activity maps that can be efficiently computed and combined. Future work will focus on automatic selection of multiple maps for high-level segmentation.

REFERENCES

[1] M. B. J. Ali, "Background noise detection and cleaning in document images," *Proc. Int. Conf. on Pattern Recognition (ICPR),* Vienna, Austria, August, 1996, pp. 758–762.

[2] R. L. de Queiroz and R. Eschbach, "Segmentation of compressed documents," *Proc. IEEE ICIP*, Washington, DC, Oct. 1997, pp. 26–29.

[3] J. Liang and R. Haralick, "Document image restoration using binary morphological filters," *Proc. SPIE Document*

Recognition III, San Jose, CA, 1996, vol. 2660, pp. 274–285.

[4] K. Chinnasarn, Y. Rangsanseri, and P. Thitimajshima, “Removing salt-and-pepper noise in text/graphics images," *Proc. Asia-Pacific Conf. on Circuits and Systems (APCCAS),* Chiangmai, Thailand, 1998.

T. M. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. Cambridge, MA: The MIT Press, 1990, p. 448.

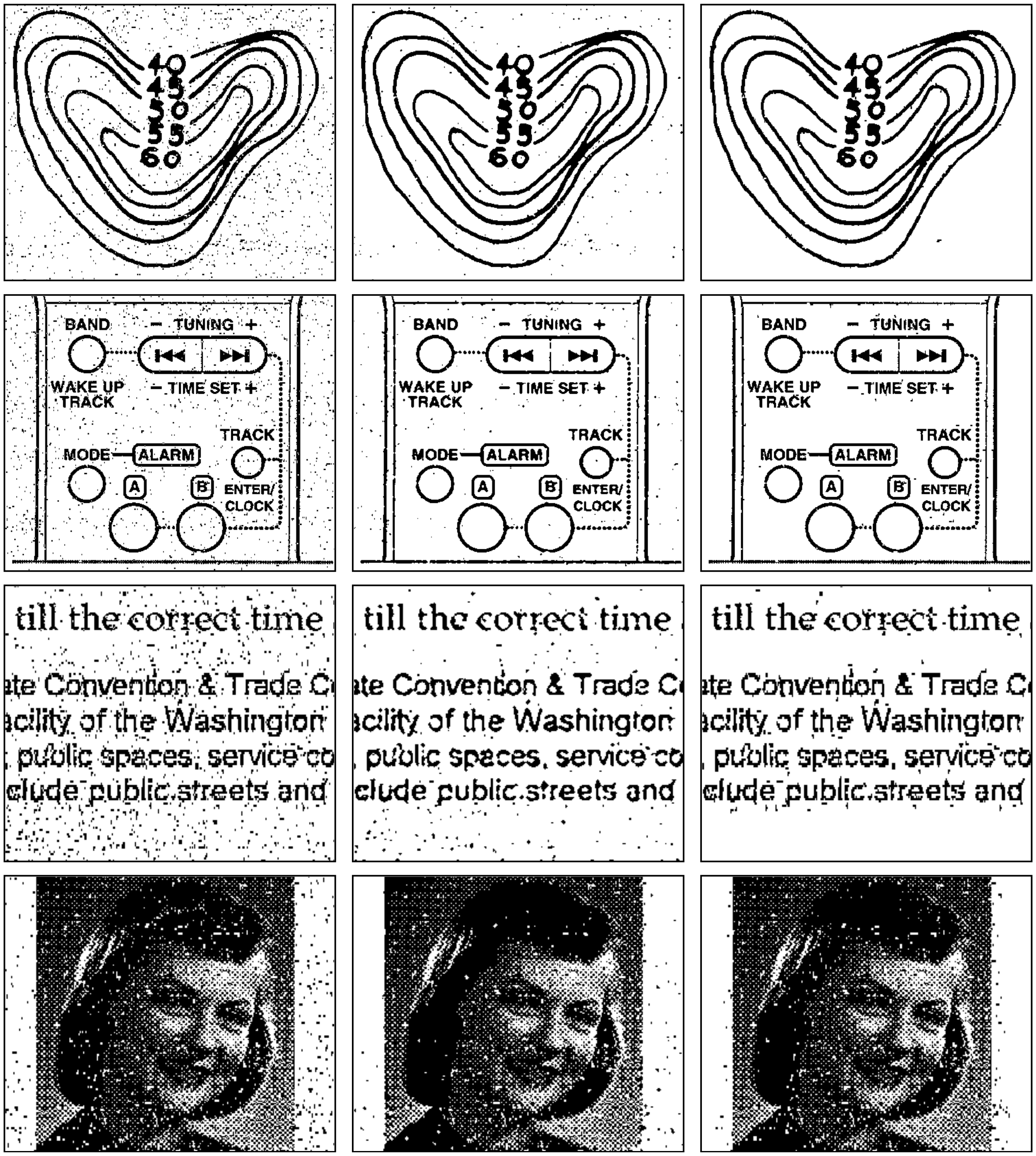


Figure 3. Noise removal results. Left column: portions of the original noisy image; center column: output of the median-based filter; right column: output of the CC-based filter.