H-0275 (H1001-009) January 12, 2010
Computer Science

IBM Research Report

Screen Processing: Detection and Segmentation of Antialiased Text

Boaz Ophir, Amir Geva, Ella Barkan, Mattias Marder
IBM Research Division
Haifa Research Laboratory
Mt. Carmel 31905
Haifa, Israel

Screen Processing: Detection and Segmentation of Antialiased Text

Boaz Ophir, Amir Geva, Ella Barkan, Mattias Marder
IBM Haifa Research Lab
boazo@cs.technion.ac.il, {geva,ella,mattiasm}@il.ibm.com

Abstract

Screen shot images is of interest for various applications. Handling of these images poses several new challenges compared to images traditionally handled by optical character recognition (OCR). One such challenge is caused by text antialiasing, where the character images are blurred by a low pass filter thus changing their appearance according to context, and fuzing character images together. In this work we offer simple low cost procedures to locate antialiased text in a screen image and to segment the text into separate character images, as a preprocessing stage to OCR. An emphasis is given to the low cost of the algorithms, enabling us to handle large images in real time.

1. Introduction

Handling of screen images is of growing interest in the last few years. Translation applications are the most common application but there are multiple business domains where automation, through automatic usage of an applications' user interface is of interest, for example:

- Automation of business processes done by back office human operators.
- Automation of access to legacy systems for the purpose of, for example, data migration.
- Automation of software testing.

In order to be able to automate a process, state of the art software must interface with the legacy systems, either through a unique API or protocol, or by instrumenting the system (i.e., modify or insert ”spy code”) to allow the automation software to access the system's internal structure and state.

These approaches are problematic for various reasons:

- Require knowledge of the legacy system's internals.
- Time consuming, requiring a developer to create system interfaces.
- Some closed systems is not accessible in this way.
- Changing the system may affect it's behavior in unintended ways.

In order to complement this problematic approach, a visual approach may also be used. This means that instead of instrumenting the application, the application may be accessed using its user interface directly [4].

In this method, in order to read the application's user interface state, the application's window is captured and analyzed. This analysis usually involves:

- Image processing, to recognize the screen features such as lines, rectangles and circles, in order to deduce controls such as fields, check boxes et.
- Optical Character Recognition (OCR) to retrieve text, such as titles, field text, etc.

On the one hand screen images are acquired under seemingly ideal conditions with zero noise level. On the other hand the complexity of these images is limited only by the designer's imagination. The image can contain characters of any of a multitude of fonts, styles, sizes, colors etc., arranged in any particular layout.

Of particular interest to us is text antialiasing. This process, explained in detail in section 2, involves horizontal blurring of the character images. While this tends to improve (albeit subjectively) text readability for the human user, it may cause problems for OCR systems. Antialiased characters, especially in small sized fonts, tend to touch one another making contour or projection based segmentation methods inapplicable.

This article deals with locating antialiased text within a screen image and segmenting the characters correctly. Our only assumptions are that characters in a word are all the same color and that the immediate background of the word is also uniform in color. An important constraint, drawn from our targeted applications, is that a the complete screen image is processed in near real time.

1.1 Previous Work

Different approaches to to handling antialiased small sized fonts have been suggested. In [10, 9, 8] a joint segmentation and recognition approach is taken. Word images are segmented (and oversegmented) then the segments are merged and classified. This type of approach requires many classification operations (much more than the number of actual characters) and as such is too time consuming for our needs.

In [2, 3] a Hidden Markov Model (HMM) is used for joint segmentation and recognition. An HMM model is matched to features extracted from a sliding window over the word image. This approach is also overly complex for our needs, specifically since competing HMM models are needed for characters of all font types and styles.

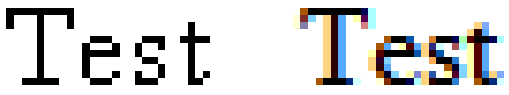
Unlike the above works our approach is based on understanding the character image creation process and using this understanding to segment the characters.

2. Antialiased Character Image Creation

We start by explaining the conceptual steps to rendering a character image. The process is composed of three stages - the character position is computed (at subpixel resolution) by the program generating the screen image, then antialiasing filtration is applied. The input to the filtration stage is a binary image with black text on white background. Finally text and background colors are applied.

Typical antialiasing filters are horizontal low pass filters. The filter can be applied either at full pixel resolution or at sub-pixel resolution according to system and program settings. In Microsoft's ClearType$^{\text{TM}}$[7, 1] the filter used is a simple 3-tap block filter with coefficients $\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\}$. This is the case we will focus on although our algorithm is not limited to this antialiasing filter.

As a result of the antialiasing process the same character can have different appearances depending on neighboring characters and on the position of the sampling grid. In particular there may be no background pixels separating adjacent characters causing well-known segmentation methods [6] used in classical OCR systems to fail.



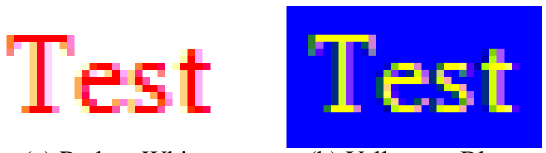
(a) No antialiasing (b) Cleartype antialiasing

Figure 1. B/W Antialiasing Example

Color is applied to the rendered characters using the following formula:

$$y = T_c + \frac{B_c - T_c}{255} x \quad , c \in \{r, g, b\} \tag{1}$$

where y is the final sub-pixel intensity, x is the sub-pixel intensity after antialiasing, T is the text color and B the background color. The different color channels do not interact so the intensity of each sub-pixel is affected only by the text and background color component that is associated with that sub-pixel position. Adjacent subpixels are thus affected differently by the coloring.



(a) Red on White (b) Yellow on Blue

Figure 2. Color Antialiasing Examples

3. Antialiased Text Detection

The first stage in handling antialiased text in a screen image is identifying the the location of the text.

Many different text detection schemes have been proposed for different scenarios [5]. None of these works however address the specific problem we are dealing with. Furthermore, such solutions do not differentiate between antialiased and non-antialiased text. Some technologies (Flex for instance) allow mixing different types of text in the same screen.

Our text detector is based on locating image areas containing the gradients caused by the antialiasing process and grouping them together into equivalence classes according to proximity.

A positive gradient detector D^p is defined on the gray level image I by:

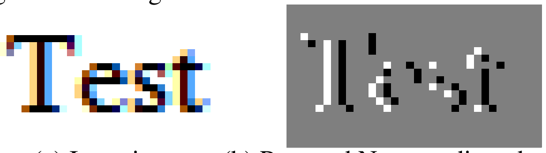
$$D^p_{(i,j)} = \begin{cases} 1 & I_{(i,j-1)} + Th < I_{(i,j)} < I_{(i,j+1)} - Th \\ 0 & otherwise \end{cases} \tag{2}$$

The operator detects pixels that are lighter than their neighbor to the left and darker than their neighbor to the right, by a threshold Th. A negative gradient detector D^n is similarly defined.

Connected components from both binary gradient detector images are then grouped into equivalence classes according to proximity. Since we expect gradients of opposite orientation when "entering" and "leaving" a character, such gradient components are grouped together if they have vertical overlap and are horizontally close. We also group together gradients of similar orientation if they have horizontal overlap and vertical proximity.

The results can further be improved by adding additional layout rules to the equivalence grouping. These rules can include looking at the image pixels between the gradients, or checking to see that an equivalence class has a base line (indicating that the class is most likely a word or sentence).

The images contained by the equivalence classes' bounding boxes are then used as input to the character segmentation stage.



(a) Input image (b) Pos. and Neg. gradient detector

Figure 3. Antialiased Text Detection

4. Character Segmentation

Our character segmentation approach is based on the understanding of the image creation process detailed in the previous section.

Our first instinct was to treat this problem as a deconvolution problem, however standard approaches such as Weiner filtering produce unstable results that are too dependent on initial and border conditions. Specifically, ripple effects can easily render the results useless.

The results of more complex deconvolution schemes, involving optimization of energy functionals and regularization terms, were likewise insufficiently accurate. In small fonts one missed pixel is enough to give incorrect segmentation results. These approaches especially suffered when text and background shared a common color component (or two).

In light of this we propose first a simple scheme for graylevel text/background segmentation and then extend it for color. It should be noted that when sub-pixel antialiasing is applied to graylevel text & background, the resulting image is actually a color image with red and blue hued pixels as well as gray pixels.

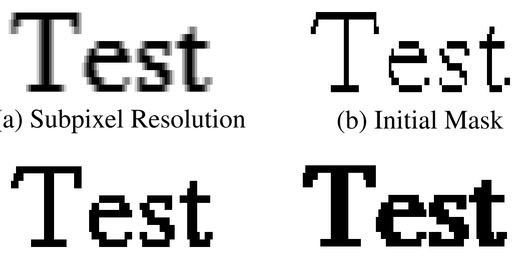
Graylevel Combining the antialiasing filtering scheme with the prior knowledge that character edges are generally sharp leads us to a very simple character segmentation approach. Character masks are grown from local minima (assuming black text on white background) following the gradient, until reaching either background or local maxima.

Character Segmentation Algorithm:

1. Sub-pixel image - for sub pixel antialiasing (clear type) create a grey level image with sub pixel resolution. Convert each RGB pixel into 3 adjacent grey level pixels, the color intensity is used as the grey level intensity. Care needs to be taken with the spatial order of the color channels (RGB or BGR). If no sub pixel antialiasing is used skip this step.
2. Contrast stretching - linearly stretch contrast to full dynamic range. In the standard case of dark text on bright background designate TEXT = 0, BACKGROUND = 255
3. Brighten local maxima to BACKGROUND value. A local maxima is defined as either:
   (a) A pixel that is brighter than both adjacent pixels (right and left).
   (b) A pixel that is brighter or equal to both adjacent pixels and brighter than both pixels 2 positions removed.
4. Initialize text mask to indicate all pixels with value TEXT or are local minima. A local minima is defined in similar manner as in the previous step.
5. Grow masks - scan the image and mask twice - one left-to-right scan and one right-to-left scan. For each location - if the current location is adjacent to a mask pixel (but not in the mask itself) then check if this location is on a gradient - if so then add it to the mask. Updates are done in place. A pixel is considered "on" a gradient if both conditions apply (this is for the case the current pixel is left of a mask pixel):
   (a) The pixel is lighter or equal to the mask pixel (which is its neighbor on the right).
   (b) The pixel is darker than its neighbor on the left either once,twice,three or four times removed.

   When the mask neighbor is on the left, switch direction of the gradient check. At the end of this stage each 4-connected component in the mask image should represent the sub-pixels associated with one character.

Working at sub-pixel resolution alleviates the problem of character appearance (at full pixel resolution) being dependent on position relative to the sampling grid.



(a) Subpixel Resolution (b) Initial Mask

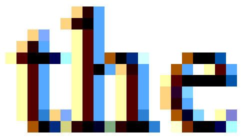
(c) Mask Growing L-R (d) Mask Growing R-L

Figure 4. Graylevel Segmentation Steps

The algorithm allows character masks to grow horizontally. Local maxima prevent mask segments in the same row from merging, however segments in adjacent rows can merge. In most cases this is a good thing since these segments usually belong to the same character, however in some cases different characters can merge. Many of these cases can be identified however since the connection is usually weak. We define a mask location (i, j) as a potential breakpoint if ALL the following conditions are fulfilled:

1. The location (i, j) is on the mask
2. The graylevel value of the sub-pixel image at location (i, j) is above a certain threshold (i.e., this pixel is rather bright).
3. The location (i, j) is on the mask border, i.e. one of it's neighbors (right or left) is NOT in the mask
4. Conditions 1-3 also all apply also to the position below, $(i + 1, j)$.
5. Positions (i, j) and $(i + 1, j)$ are mask borders from opposite sides, i.e. if the right side neighbor $(i, j+1)$ is not in the mask and neither is the left side neighbor below $(i + 1, j - 1)$ (or vice versa).

We used a threshold value of 0.5, equivalent to 127 on the contrast stretched sub-pixel image.


(a) Input image

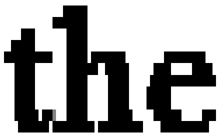

(b) Mask w/ breakpoint (gray)

Figure 5. Candidate Breakpoint

Color Assuming the colors of both text and background are known, we can invert equation (1) to recover x and achieve "decoloring":

$$x = \frac{255}{B_c - T_c}(y - T_c) \quad , c \in \{r, g, b\}. \tag{3}$$

After decoloring the segmentation algorithm can be applied as is.

This method is susceptible to two difficult scenarios:

- A color component is identical in both Text and Background. In this case recovering x directly is impossible.
- Color saturation at character borders can cause text sub-pixels to merge with the background (and vice versa).

Several simple procedures, augmenting the decoloring stage, can handle most of these cases:

- If the whole pixel (all 3 color channels) is equal to the Background (Text) color, designate its sub-pixels as Backround (Text) and give them value 255 (0).
- If a neighboring pixel (3 sub-pixels) is equal to the Background (Text) color, designate the current sub-pixel as Backround (Text) and give it value 255 (0).
- In all other cases - interpolate between nearest recoverable sub-pixels.

While this segmentation algorithm is very simple and effective, it is not 100%. There are instances that any combination of the above still give mistaken results at character borders:

- A text sub-pixel is categorized as background leading to over segmentation.
- A background sub-pixel is categorized as text leading to under segmentation.

In addition, even without antialiasing, in some combinations of characters, font and font size, characters are fuzed together with no background subpixel between them. In these cases much more expensive joint segmentation-recognition approaches need be applied.

5. Results

The evaluation of the quality of our approach is done at this time by human experts. An automated system, including OCR, is currently under development. The detection algorithm was tested on a benchmark of screen images. Almost all the antialiased text was successfully detected with the exception of some isolated characters (not enough hits to create a meaningful equivalence class). In addition, some text adjacent or overlapping with screen graphics was grouped together with some part of graphics. The segmentation algorithm was tested on text only images generated by word processing software for various fonts, sizes and colors (both text and background). Results were generally good, with a large percentage of the characters segmented correctly. In some instances all the characters were segmented correctly with the exception of characters that were fuzed together prior to antialiasing. The results were understandingly degraded for very small fonts and text/background color combinations with some similar color channels.

6. Conclusion and Future Work

In this paper we introduced low cost procedures for detection and segmentation of antialiased text in screen images, with good results. These procedures can be used with low overhead as preprocessing stages prior to OCR. We are currently integrating the detection and segmentation with a fast OCR engine optimized for screen images. Future work will focus on adding an adaptive element to the OCR engine designed to handle segmentation/matching errors.

References

[1] C. Betrisey, J. Blinn, B. Dresevic, B. Hill, G. Hitchcock, B. Keely, D. Mitchell, J. Platt, and T. Whitted. Displaced filtering for patterned displays. In *Proc. Soc. for Inf. Display Symposium*, pages 296–299, 2000.

[2] F. Einsele, R. Ingold, and J. Hennebert. A hmm-based approach to recognize ultra low resolution anti-aliased words. In *Int. Conf. on Pat. Rec. and Mach. Int.*, 2007.

[3] F. Einsele, R. Ingold, and J. Hennebert. A language-independent, open-vocabulary system based on hmms for recognition of ultra low resolution words. *Journal of Universal Computer Science*, 14(18):2982–2997, 2008.

[4] A. Geva and E. Walach. US 2008/0001959 A1: System, method and computer program product for performing information transfer using a virtual operator. US Patent Application, Jan. 2008.

[5] K. Jung, K. Kim, and A. Jain. Text information extraction in images and video: a survey. *Pattern Recognition*, 37(7):977–997, 2004.

[6] G. Nagy. Twenty years of document image analysis in PAMI. *IEEE Trans. on Pat. Anal. and Machine Int.*, 22(1):38–62, 2000.

[7] J. Platt. Optimal filtering for patterned displays. *IEEE Signal Processing Letters*, 7(7):179–181, 2000.

[8] S. Wachenfeld, S. Fleischer, and X. Jiang. A multiple classifier approach for the recognition of screen-rendered text. In *Proc. of Int. Conf. on Computer Analysis of Images and Patterns*, pages 921–928, 2007.

[9] S. Wachenfeld, H. Klein, S. Fleischer, and X. Jiang. Segmentation of very low resolution screen-rendered text. In *Proc. of Int. Conf. on Doc. Anal. and Rec.*, 2007.

[10] S. Wachenfeld, H. Klein, and X. Jiang. Recognition of screen-rended text. In *Proc. of the 18th Int. Conf. on Pattern Recognition*, volume 2, pages 1086–1089, 2006.