

Using Computer Vision to Detect Web Browser Display Errors

Xu Liu, David Doermann

Institute for Advanced Computer Studies

University of Maryland, College Park, Maryland 20742, U.S

{liuxu,doermann}@umiacs.umd.edu

Abstract

As the functionality and complexity of the WWW continues to grow so does the need for WWW quality assurance and testing. Although there have been numerous approaches to automated Web testing, existing techniques analyze primarily textual information, and the final judgment on correctness of layout is via human observation. The motivation of this paper is to employ computer vision techniques to detect Web display errors. To do this, we analyze images of the rendered pages rather than the HTML to discover errors. Our approach includes page segmentation, dynamic matching and outlier identification. We show that the approach successfully detects layout errors in the Opera browser on Microsoft Websites, while minimizing false alarms.

1. Introduction

When users browse the internet, they occasionally see unpleasant displays which are not the intension of the Web designer. Web errors (including inaccessibility, slow speed, and display errors) cost businesses in the UK £565 million in 2001[5]. Consequentially, Web quality assurance and testing are required to prevent these errors before they are deployed to the Web server. Ideally we would like to automatically detect Web display errors before they are displayed to users. Although Web testing techniques exist, they focus primarily on textual information. A statistical approach for Web log analysis is studied in [1]. Ricca, et al [2] proposed dynamic testing driven by UML. Elbaum, et al [3] utilizes user session data for Web testing. These techniques, however, are not based on the information that the users really “see”, since they are all measured prior to rendering. Text information, in fact, is not sufficient because the HTML may be rendered differently by different browsers. Some common cases include:

(1) Different HTML parses

Internet Explorer, Firefox, Mozilla, Netscape Navigator and Opera each have their own HTML parsers, and although these parsers generally render the same result for the same HTML, they may differ in

details causing display variance and bugs. Even those browsers which use Internet Explorer as a COM component, such as the Maxthon or Avant Brower, disagree on certain pages because of security settings.

(2) New Web directives

Web developers are always eager to use cutting edge Web techniques, writing fancy DHTML/CSS directives may result in impressive displays on certain browsers but not be rendered correctly by others or earlier versions.

(3) Browser fault tolerance

Internet Explorer has strong error tolerance so that HTML bugs may not appear but they will appear on other browsers which are less tolerant.

(4) Browser extensions

Microsoft has added many HTML extensions to Internet Explorer and FrontPage, however these extensions are not included in the W3C standard. HTML with extended scripts cannot be parsed correctly by browsers which strictly follow the W3C standard. Firefox also has extension packages but without installing these packages, some pages cannot be displayed correctly.

Hence, analysis of the image of the rendered page is required for post-browser Web quality assurance and testing. Since 2001, the WDA has brought together researches that support image and text based Web document analysis. Antonacopoulos, et al [4] used human perception cues and fuzzy inference to extract text from Web images which is an initial approach to segmentation and extraction. To our knowledge there has not been any image based Web testing approach proposed.

By analyzing the four reasons for Web display errors listed above, we find that the main challenge is in the variety of browsers. Our assumption is that most of the browsers render pages correctly, and outliers are most likely errors. Thus our goal is to find the outlier pages among normal pages. The remainder of this paper is organized as follows: Section 2 defines the problem and outlines our general solution. Section 3 discusses the algorithm in detail. Section 4 highlights the experimental results and conclusions and future work are presented in Section 5.



Figure 1 – overlapping text

2. Problem Definition

Figure 1 shows a typical text overlapping error in Opera 7.54. Basically we have three kinds of Web display errors:

- (1) Text or image overlap

An image or text area appears on top of another image or text area making both of them unrecognizable.

- (2) Inserted Space

Web designers may use a blank image to occupy certain area to maintain alignment. However browsers behave differently with respect to blank areas. Making them either too wide or too thin can disturb the alignment of the page.

- (3) Missing text/images

For security reason, a browser can block Java Script, ActiveX and Flash plug-ins. To maintain the general layout of page, browsers will often put an “x” instead of the image.

Our task is, for a collection of Web pages, find outliers which contain errors and locate the errors on the page. To do this, we first define a measure of the difference between two pages P_1 and P_2 . We have found that the three kinds of errors listed above are all local errors, i.e. the page with display error looks good in general, though a part of it may be incorrect. This suggests that we cannot compare two images globally. Global comparison may neglect those local errors, and even if the errors are detected, global comparison can not tell exactly where the errors are. Consequentially we have to segment the pages first. By segmenting pages P_1 and P_2 into segments $P_{1,1}, P_{1,2} \dots P_{1,m}$ and $P_{2,1}, P_{2,2} \dots P_{2,n}$, we can define the measurement of difference between P_1 and P_2 as:

$$D(P_1, P_2) = \sum_{i=1..m} \min_{j=1..n} D'(P_{1,i}, P_{2,j}) \quad (1)$$

Here D' is the difference between two segments. With (1) we have simplified the page comparison to segment comparison and finding the optimal match can be accomplished with dynamic programming.

For a collection of web pages $P_1, P_2 \dots P_n$ we define the average difference from one page P_i to all the other pages as:

$$\bar{D}(P_i) = \frac{\sum_{j=1..n, j \neq i} D(P_i, P_j)}{n-1} \quad (2)$$

Thus, those pages with a high \bar{D} value are probably outliers because they are distinct from most of the others.

3. Algorithm

General image segmentation has been intensively studied in the literature. Early segmentation algorithms use color and texture [6]. In 1997, normalized cut [7] was introduced to image segmentation and “fast image segmentation” [8] appeared later. However, none of these approaches produces satisfactory results, lagging far behind human ability. Although image segmentation remains an essential requirement for computer vision, existing algorithms may not be appropriate for Web document images because of their graphical nature and may not provide correct result for artificial images like Web images and they are often slow. Figure 2 shows segmentation result of “Normalized Cut [7]” on a partition of a Web page.

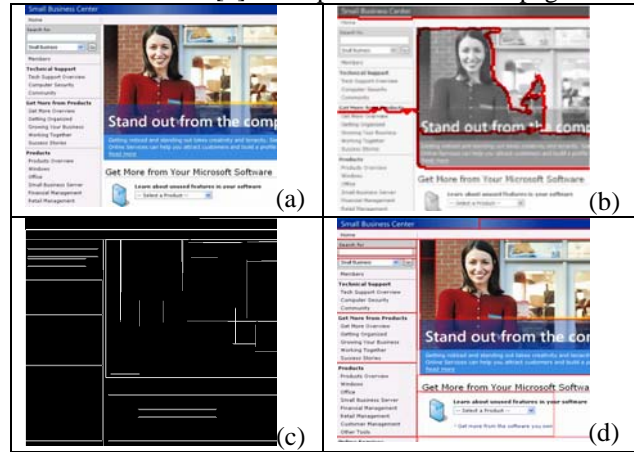


Figure 2 Page Segmentation

Normalized Cut [7] requires 30 seconds on Pentium M 2.0GHz CPU and yields a segmentation result as shown in Figure 2(b). We can see that the segmentation is incorrect as segments on the bottom and left are not segmented and the image on top right is over segmented. General image segmentation algorithms like Normalized Cuts fail on web document images because they seek a global optimization minimizing energy under certain criteria. However the global solution always biases the salient parts of images. This does not mean, however, that Web page

segmentation is unsolvable. The particularity of Web images provides us clues to design the Web page segmentation algorithm: Web page contains text and images which are often horizontal; hence, Web pages can often be segmented into rectangles.

Our page segmentation algorithm contains three steps

(1) Edge detection

In this step we use the Sobel[9] edge detector to find edges. We have implemented Prewitt, Canny[10] and Susan edge detectors and found that Sobel detector produces the best results. After edge detection, only the vertical or horizontal lines that are long enough are retained. Most of the short noise is removed, thus the computational complexity is largely reduced. Figure 2(c) shows an edge detection result.

(2) Over segmentation

Every line remaining after edge detection is a cut point. Vertical lines cut the page into left and right regions while horizontal lines cut the page up and down. Eventually the page is cut into a lattice, on which we enumerate every possible rectangle. For each rectangle, we calculate the consistency of texture inside it. If the consistency is high enough, it is considered a salient block and is segmented from the lattice.

(3) Merging

After over segmentation, the page is segmented into rectangles. Neighbor rectangles merge into one if their colors are similar.

Result of page segmentation is shown in Figure 2 (d). We now have the segments $P_{1,1}, P_{1,2} \dots P_{1,m}$ and $P_{2,1}, P_{2,2} \dots P_{2,n}$, and we need to calculate $D(P_{1,i}, P_{2,j})(i = 1..m, j = 1..n)$, i.e. to compare each pair of segments. The naive approach to comparing two images is pixel by pixel comparison, but it is not feasible because missing one line will result in huge difference which is not our intension.

<p>Get More from Products</p> <p>Get More Overview</p> <p>Getting Organized</p> <p>Growing Your Business</p> <p>Working Together</p> <p>Success Stories</p>	<p>Get More from Products</p> <p>Get More Overview</p> <p>Getting Organized</p> <p>Growing Your Business</p> <p>Working Together</p> <p>Success Stories</p>
(a) Text image in IE6	(b)Text image in FireFox 1.0

Figure 3 Same Texts in Different Browsers

Figure 3 shows two images of the same texts rendered by different browsers. Notice that the one in FireFox is wrapped thus longer than the one in Internet Explorer, though they should be considered the same.

Dynamic Warping (also known as Dynamic Time Warping, DTW) [11] is a mature algorithm for signal matching, especially in the presence of distortion. Rath [12] uses Dynamic Warping for matching hand-written word images. To apply Dynamic Warping to comparing images we first convert the 2D image to 1D signal by calculating the average color of each line.

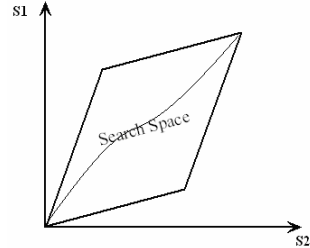


Figure 4 Dynamic Warping

Figure 4 shows a typical Dynamic Warping search space and a Dynamic Warping path. On The Dynamic Warping path, one signal is squeezed and the other is stretched. Dynamic Warping searches every possible path in the search space to find the minimum difference between S_1 and S_2 . By dynamic programming, those steps that have been calculated do not need to be recomputed, thus the complexity is $O(mn)$, where m, n are the lengths of S_1, S_2 respectively. To calculate the difference between S_1 and S_2 , we iteratively compute

$$\left. \begin{aligned} Diff(0,0) &= 0 \\ Diff(i,j) &= \min_{i1 < k < i2} \{ Diff(k, j-1) + |S_1(i) - S_2(j)| \} \\ |S_1 - S_2| &= Diff(S_1.length(), S_2.length()) \end{aligned} \right\} (3)$$

Applying Dynamic Warping to Figure 3 yields very few differences. Those segments in one browser that have no match in other browsers have a high possibility of error.

4. Experiments

We applied our algorithm to 40 pages rendered by Microsoft Internet Explorer 6, Mozilla 1.7.3, Firefox 1.0 and Opera 7.54. We have found that Opera 7.54 has layout bugs on most of the Microsoft Websites.

Table 1 shows a comparison between segments of one page rendered by IE ($P_{1,1}, P_{1,2} \dots P_{1,19}$) and Opera ($P_{2,1}, P_{2,2} \dots P_{2,13}$) respectively. Before running our algorithm we manually put /// in the cells which are known to be the same segment, and our algorithm

results in differences (typically < 100) in these cells which are consistent to our prior knowledge.

	P _{1,1}	P _{1,2}	P _{1,3}	P _{1,4}	P _{1,5}	P _{1,6}	P _{1,7}	P _{1,8}	P _{1,9}	P _{1,10}	P _{1,11}	P _{1,12}	P _{1,13}	P _{1,14}	P _{1,15}	P _{1,16}	P _{1,17}	P _{1,18}	P _{1,19}
P _{2,1}	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
P _{2,2}	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	44	∞	∞	16	∞	47	∞	∞	∞
P _{2,3}	∞	∞	129	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
P _{2,4}	∞	∞	∞	∞	∞	30	∞	∞	∞	∞	∞	∞	2999	∞	∞	∞	∞	∞	∞
P _{2,5}	∞	∞	∞	50	16	∞	0	∞	100	122	247	92	∞	187	125	∞	116	∞	0
P _{2,6}	∞	∞	∞	73	126	∞	138	∞	139	21	35	20	∞	27	20	∞	19	∞	30
P _{2,7}	∞	54	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
P _{2,8}	∞	∞	∞	∞	∞	∞	∞	69	∞	∞	74	∞	∞	109	∞	50	∞	∞	∞
P _{2,9}	∞	∞	∞	∞	∞	∞	∞	119	∞	∞	∞	∞	∞	∞	∞	∞	∞	37	∞
P _{2,10}	∞	∞	∞	∞	∞	3008	∞	∞	∞	∞	∞	∞	44	∞	∞	∞	∞	∞	∞
P _{2,11}	∞	∞	∞	11	20	∞	∞	∞	100	33	247	22	∞	187	30	∞	29	∞	0
P _{2,12}	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
P _{2,13}	∞	∞	∞	115	203	∞	∞	∞	197	33	37	70	∞	71	40	68	51	∞	196

Table1- Page Comparison

Before running Dynamic Warping on each pair of segments we filter out the obvious differences, e.g. a 10x10 segment is definitely different from a 20x70 segment as is a red segment and a blue segment. Those cells with ∞ are filtered out. Those rows which have no // cell, i.e. every value in the row is bigger than a threshold (typical 100), is considered an outlier because it looks far more different than others. In this table, P_{2,3} and P_{2,12} are probably outliers because they have no match among P_{1,1}, P_{1,2} ... P_{1,19}.



Figure6

<http://www.microsoft.com/learning/default.asp> by Opera 7.54 one image is separated and a blank rectangle is inserted

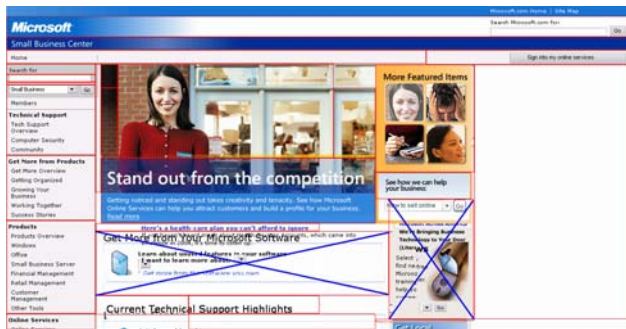


Figure5

<http://www.microsoft.com/smallbusiness/default.ms> p by Opera 7.54 two blocks have text and image overlap error

We ran our algorithm on 20 pages rendered by 5 different browsers and successfully identify the 3 pages which contain display errors. Figures 5 and 6 show two typical errors. Those blocks with error display in these pages are marked with a blue cross. In fact, Opera 7.54 has display errors on most of the pages under www.microsoft.com and can be detected by our program.

5. Conclusion and future work

To detect Web browser display errors we proposed a solution which consists of two parts – segmentation and matching. The segmentation is based on Sobel edge detection and the matching uses Dynamic Warping. Our algorithm successfully detects errors in one Web browser. However the page segmentation is not very accurate and relies on the saliency of page blocks, thus it yields false alarms. Our future work is to improve the accuracy of segmentation and to reduce the false alarm. Also, the images being tested are captured manually. Our future work includes making the capturing and testing completely automatic, so that we can testify our approach on a large number of pages.

6. References

- [1] Kallepalli, C. Tian, J. Iris Labs Inc, Plano, TX; Measuring and modeling usage and reliability for statistical Webtesting IEEE Trans. on Software Engineering, Nov. 2001.
- [2] F Ricca, P Tonella, Analysis and Testing of Web Applications, ICSE, 2001
- [3] Elbaum, S. Karre, S. Rothermel, G. Improving Web application testing with user session data Proceedings. ICSE, 2003.
- [4] A. Antonacopoulos and D. Karatzas, Text Extraction from Web Images Based on Human Perception and Fuzzy Inference, WDA 2001
- [5] Philip Sheldon, Basic Errors Plague E-Commerce Sites, ECOMMERCE IN ACTION, 05/29/2002
- [6] B.S. Manjunath and R. Chellappa. Unsupervised texture segmentation using markov random field models. IEEE Trans. on PAMI 13:478--482, 1991.
- [7] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. IEEE Trans. on PAMI, 22(8):888-905, August 2000.
- [8] Eitan Sharon, Achi Brandt, and Ronen Basri, Fast Multiscale. CVPR2000. South Carolina, Vol. I: 70-77, 2000.
- [9] R. Gonzalez and R. Woods Digital Image Processing, Addison Wesley, 1992, pp 414 - 428.
- [10] R. Boyle and R. Thomas Computer Vision: A First Course, Blackwell Scientific Publications, 1988.
- [11] C. S. Myers and L. R. Rabiner. A comparative study of several dynamic time-warping algorithms for connected word recognition. The Bell System Technical Journal, 60(7):1389-1409, September 1981.
- [12] Rath, T.M. Manmatha, R. Word image matching using dynamic time warping, CVPR 2003

Microsoft has copyright to the web pages under www.microsoft.com which are cited in this paper.