Vision Based Page Segmentation Algorithm: Extended and Perceived Success

M. Elgin Akpınar[1] and Yeliz Yeşilada[2]

[1] Middle East Technical University
Ankara, Turkey
[2] Middle East Technical University
Northern Cyprus Campus
Mersin 10, Turkey
{elgin.akpinar,yyeliz}@metu.edu.tr

Abstract. Web pages consist of different visual segments, serving different purposes. Typical structural segments are header, right or left columns and main content. Segments can also have nested structure which means some segments may include other segments. Understanding these segments is important in properly displaying web pages for small screen devices and in alternative forms such as audio for screen reader users. There exist different techniques in identifying visual segments in a web page. One successful approach is Vision Based Segmentation Algorithm (VIPS Algorithm) which uses both the underlying source code and also the visual rendering of a web page. However, there are some limitations of this approach and this paper explains how we have extended and improved VIPS and built it in Java. We have also conducted some online user evaluations to investigate how people perceive the success of the segmentation approach and in which granularity they prefer to see a web page segmented. This paper presents the preliminary results which show that, people perceive segmentation with higher granularity as better segmentation regardless of the web page complexity.

Keywords: Web Accessibility, Web Page Segmentation, Reverse Engineering, User Study.

1 Introduction

Web pages are typically designed for visual interaction. They include many visual elements such as header, footer, menu, etc that guide the reader. One can easily look at the visual rendering and can differentiate the segments which typically differs in background color, font styles, borders or margins around the segments. On the other hand, the underlying source code typically does not provide such kind of clear segmentation or pattern. Therefore, when web pages are automatically processed by assistive technologies or adapted for mobile devices this kind of information is not available.

In order to address this problem, there exist different techniques for identifying visual segments in a web page [3,1,7]. These approaches differ in the techniques they use for automating the process, some use machine learning techniques, some uses heuristics, etc. Once a web page is automatically segmented, web pages are then typically

Q.Z. Sheng and J. Kjeldskov (Eds.): ICWE 2013 Workshops, LNCS 8295, pp. 238–252, 2013.

re-engineered for better presenting them on mobile devices [28,12,19] and for screen reader users [14,4,27]. Reengineering approaches include variety of techniques, for example allowing user to directly read the main content, or allowing user to skip the menu or diving a web page into several small pages, etc. (see Section 2).

A well known segmentation technique is Vision Based Segmentation Algorithm (VIPS Algorithm) [7]. VIPS uses both the underlying source code and also the visual rendering of a web page. This technique is very popular and has been used to segment web pages for mobile devices as well as improving the precision and recall of information retrieval results. When a web page is segmented by VIPS, it generates a block tree of a web page. This tree is structurally similar to the underlying DOM tree, but it includes the hierarchy of visual elements on web pages. Nodes in the top levels represent higher level segments and nodes in the lower levels include the lower level, higher granularity segments. This approach therefore gives a segment tree that can be used in different levels (see Section 3).

Even though VIPS is proven to be a very successful approach [7], there are some limitations. These limitations consist of poorly defined order of precedence for the rule set and tresholds which are applied on DOM elements in segmentation and adatibility problems to new technologies such as HTML 5 and dynamic web contents. This paper explains how we have extended and improved VIPS and built it in Java (see Section 4). We have also conducted an online user evaluation to investigate how people perceive the success of the segmentation approach and in which granularity they prefer to see a web page segmented. In particular, we have investigated the following two research questions: (1) What is the perceived success of our extended segmentation algorithm? How does this differ based on the web page complexity? ; (2) Which level of segmentation is the most preferred?. This paper presents the preliminary results which show that, people perceive segmentation with higher granularity as better segmentation regardless of the web page complexity (see Section 5). This finding is very important for being able to decide how to segment a web page automatically and display it to mobile and disabled users for better interaction experience.

2 Related Work

Web page segmentation has been primarily used to re-engineer pages so that they are easier to access by mobile web users [28,12,19] and by visually impaired users who access web pages in audio [14,4,27]. Our literature review also shows that the segmentation process has been proposed and used in different fields, for example, information retrieval [7,6], image retrieval [5], etc. Here, we briefly present related work in mobile web and web accessibility fields.

There are still many problems associated with mobile web access. Web pages are typically designed for certain screen sizes [11,28] and therefore scaling down for small screen devices or scaling up for large displays can be difficult [16], some mobile devices also do not have keyboard or mouse [24], they have limited processing capabilities [24] and they tend to have problems with multimedia data [12,19]. Furthermore, the information need is very different for mobile web users, people focus more on getting direct answers to specific questions, and expect more relevant and clear results instead of

browsing through large amount of data [24]. Most important of all of these is that the relationship between web page segments and elements are not semantically explicit so one cannot easily display a page on another or alternative platform [20,28]. Therefore, segmentation process provides a good way of identifying segments in a web page such that they can be used to better display a web page to mobile and screen reader users.

When we look at the existing work, there are different approaches that make use of segmentation for reengineering web pages. [28] aims to extract and present only the important parts of the web page for mobile web access. [17] similarly aims to discover the importance of segments in a web page – block-importance information can be used to decide which part of the page need to be displayed first to the user. [29,24] focus on identifying the main content of the page so that it is directly presented to the user. [1,22,10,9] split a web page several pages. [1] aims to identify information blocks so that a web page can be divided into smaller pages and represented to the user with a table of contents. [12,19] propose a reengineering technique called the indexed segmentation, which transforms segments or components of web pages into a sequence of small sub-pages that fit the display of a hand-held device, and binds them with hyperlinks. [11] proposes to segment the web page into several smaller pages and then displays a table of contents called "object lists" and provides a link to each segment from this table of contents.

There are also some other work that uses segments to provide zoom in and out of a web page [16], identify the navigation and content blocks, and then filter the relevant ones and show the summarizes to the user[1], identify the segments and filter out the irrelevant or unnecessary ones [22], identify the navigation bar and do not display it if the page is a content page [8]. [8] similarly proposes to remove decoration and special objects such as adverts, logo, contact, copyright, reference, etc. [12,19] propose a transcoding technique for outlining the sections by using the section headers. In summary, the page is kept the same but the section header is converted to a link that points to the content of that section. Highlights the importance of blocks/segments in a web page with different color. It shows a thumbnail view of the page with the importance of the blocks highlighted [24].

[21] presents the complete web page to the user and the user then clicks on regions on the web page to retrieve sub-pages. Similarly, [23] aims to show the user the thumbnail view of the page and when the users move their mouse over a section, they asynchronously retrieves that block and displays it to the user. [10,9] also aim to present the thumbnail view of the page and then the user can move to sub-pages. [25,3] present the thumbnail view of the page and the full screen information about a segment, the user can move from one block to another to retrieve the content.

When we look at web accessibility field, we also see that visually impaired users also experience similar problems to mobile web users and similar approaches have also been proposed for web accessibility [26].

When the page is visually segmented but this is not encoded in the source code, applications such as screen readers cannot access that information. Therefore, the page becomes very inaccessible to screen reader users. [2,18] aim to identify visual segments in a web page such that the page can be re-engineered to better support accessibility. They propose to manually annotate the page to identify the role of segments in a page.

Similar approach is proposed by [27] in the Dante project, where the visual segments in a web page is annotated by an ontology and that ontology is then used to re-engineer web pages for visually impaired web users. Similar approach has the been proposed by [13] where the segmentation of visual elements have been automated by using the underlying CSS of a web page. [14,4] indicates that the applications such as screen readers process a web page sequentially (i.e., they first read through menus, banners, commercials, etc) therefore this makes browsing time-consuming and strenuous for screen reader users. Therefore, with the specialized audio browser called Hearsay or CSurf what they try to do is that when the user clicks on a link they aim to retrieve the target page and point the user to the relevant block to that link. They do this by segmenting a web page into several blocks and then identifying the context and the relevant block to that link.

When we investigate the approaches used to segment a web page, we see that they can be grouped into two: top-down and bottom-up page segmentation. Top-down web page segmentation approach starts with the root node of a page and segments this node into smaller blocks iteratively, using different features obtained from the content of the page [3]. Bottom-up web page segmentation approach, on the other hand, starts with atomic content units such as the leaf nodes of the DOM representation [3]. Different algorithms have been also proposed to automate the process of segmentation which includes heuristics approach [1], machine learning [3], clustering, etc. One successful approach is the VIPS algorithm that combines the analysis of both the underlying source code and the visual rendering of a web page [7]. However, there are some limitations of this approach. Therefore, in our work we have extended VIPS. In the following sections, we first introduce VIPS and then discuss our improvements.

3 VIPS: A VIsion Based Page Segmentation Algorithm

Web pages are based on a hierarchical DOM structure which consists of both significant and insignificant nodes having contextual or representational purpose in the page layout. DOM structure includes all the nodes and their relationships in a web page. Visual rendering of the page, on the other hand, eliminates insignificant nodes and restructure the page layout with respect to the representation of the nodes. It helps to disregard invisible nodes, differentiate various tag types and group bulk content by using visual attributes, such as font size, background color or margin. Most segmentation algorithms focus on the DOM structure but visual rendering is also an important factor in segmentation for differentiating blocks of visual elements in a web page.

Figure 1 represents two segmentation approaches. The first in Figure 1.a is based only on the DOM structure of the page. The figure shows that, it highlights invisible nodes as content blocks and disregards some visual cues, such as background color, which are used to group content. For example, although DB.3 is invisible and insignificant, it is in the content structure as it appears in the DOM structure. Moreover, DB.4 and DB.5 have the same background color, which is different from the background color of both DB.1 and DB.2. The color differentiation implicitly indicates that they are related, but DB.4 and DB.5 are represented as different blocks. In combination of both DOM structure and visual rendering, on the other hand, provides much richer information and therefore provides better segmentation.

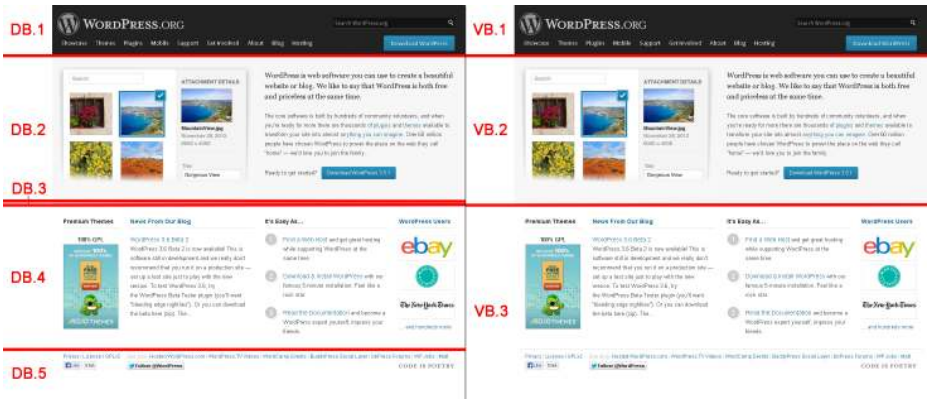

Fig. 1. 1st level of segmentation based on a) DOM structure (left) and b) both DOM and visual rendering (right)

In brief, DOM structure on its own does not provide sufficient segmentation results as contains invisible nodes and it provides only structural organization of the nodes. In order to segment pages to detect coherent blocks, visual representation need to be also used in coordination with the underlying DOM structure. VIPS Algorithm achieves this by examining a set of visual cues of each node in the DOM structure. Therefore, it provides a much more convenient technique for web page segmentation. However, VIPS Algorithm still has some limitations. In the following sections, we summarize the VIPS Algorithm and explain its limitations.

3.1 VIPS Algorithm

VIPS Algorithm first labels all the nodes in a web page with respect to their visibility in the page layout, the line-breaks they produce and their children nodes [7]. It labels the nodes which are displayed in a page layout as valid nodes. It divides the blocks according to whether they produce a line-break in rendering. If a node has a HTML tag which only affects the textual appearance (such as italic or bold font) and if it is applied to a textual content without introducing a line-break, then this node is labeled as inline node. Otherwise, if the HTML tag produces a line-break before or after its representation, it is labeled as line-break node. Free text nodes, which does not have a HTML tag are labeled as text nodes. The nodes which only consist of text nodes or inline nodes are labeled as virtual text nodes.

After labeling the nodes in a web page, VIPS Algorithm aims to find visual blocks in a web page in three main steps which are: (1) visual block extraction, (2) visual block separation and (3) content structure construction. In visual block extraction part, some visual cues of the nodes are examined to decide whether create a block for the node on some predefined rules. In visual block separation, using the visual cues further, VIPS Algorithm tries to find the relevance between two blocks. Finally, in content structure construction, VIPS constructs a block tree of visual blocks in a hierarchical structure.

Visual block extraction is the major part of the algorithm, since visual segments are defined in this part. Some predefined rules are applied on nodes to decide whether to construct a block or not. This decision is mainly taken on the label of the node (valid, inline or line-break) and labels, background colors, font styles and sizes of its children. Also, in each visual block extraction step, a Degree of Coherence (DoC) value is assigned to each block. When a block is assigned bigger DoC value than a predefined value, the algorithm stops segmenting for the corresponding node. After one iteration of visual block extraction process, the algorithm returns a set of visual blocks which are related to some of the nodes in the DOM structure of the web page.

In visual block separation, the algorithm first detects the separators between the blocks extracted in the previous step. These separators are the spaces which surround the visual blocks. Then, the algorithm assigns a weight to each separator, by examining background color and font size of the blocks at both sides and height and width of the separator. This weight initiates the relevance of the content of two blocks.

In the third, all extracted visual blocks are appended to a tree of visual blocks. The root of this block tree is the visual block which represents the root node of the DOM. The algorithm jumps to visual block extraction and repeats the 3 steps for the children of nodes which examined in previous iteration. Finally, the algorithm terminates when there is no other node to examine.

At the end of the execution of the algorithm over a web page, it produces a tree of visual blocks in a hierarchical structure. This tree is different from the DOM structure of the page, since the algorithm eliminates the invalid nodes and group the adjacent virtual text nodes and inline nodes into one single visual block. Therefore, the tree of visual blocks is simpler when it is compared to the DOM structure of a web page. Figure 2 represents an example segmentation result for the Wordpress.org. Firstly, the page is divided into larger visual blocks, such as VB1.1, VB1.2, VB1.3 in Figure 2. Then, these blocks are further divided into smaller visual blocks. For instance, VB1.3 is further divided into VB1.3.1, VB1.3.2 and VB1.3.3. In our ACTF implementation, the visual element identifier generates the tree of visual elements and creates an XPATH[1] for each visual block with respect to its corresponding node.

3.2 VIPS Limitations

The algorithm is very effective since it combines both the tag attributes of the nodes and visual properties to detect the visual blocks of the page. However, this method of segmentation has several limitations. One major drawback of this approach is that, in the rule set which was defined for visual block extraction, precedence of the rules and thresholds are not well defined. Another problem with this approach is that, the tag set mentioned in the rules are narrow and does not contain HTML5 tags, and the rule set needs to be extended with additional visual attributes. Finally, VIPS has no detailed understanding and detection of dynamic contents.

Although the rule set in the study covers many visual aspects of web pages, the precedence of the rules, the application order on a node is not well defined. This causes some ambiguities in implementation of the algorithm. The order of the representation of

[1] `http://www.w3.org/TR/xpath/`

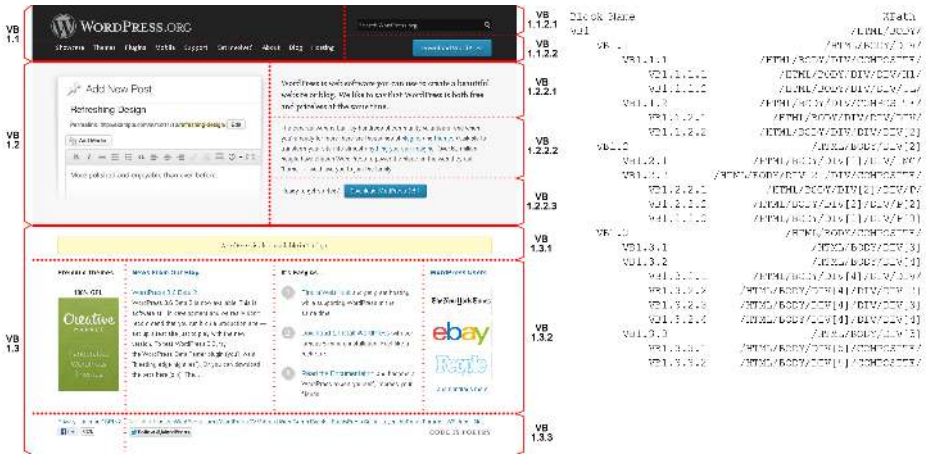

Fig. 2. Visual blocks for Wordpress.org and its corresponding block structure

the rules also conflicts with the given example in the context of visual block extraction. Moreover, the criteria of defining thresholds for size of the nodes to apply for some rules, is not also defined. Therefore, some of the rules need to be clarified, and some additional rules should be defined in the rule set.

Although the nodes are classified under categories with respect to their visual representation in a web page, HTML tags given in the study are very limited. The list of tags contains six tag types, which are inline text nodes, TABLE, TR, TD, P and other tags. However, this list is so narrow that, it does not cover neither all HTML tags nor any of HTML5 tags, which was released after VIPS was introduced [7]. Other tags than TABLE, TR, TD and P have different properties and behaviors in a web page so that they should not be used in the same category. Also, some of the tags produce unnatural visual representation when they are not used as they should be. For example, a TABLE node consists of TR nodes, which also consists of TD nodes. If we append a TD node without a TR node directly to the TABLE node, it results in a representation that does not result in a line-break. On the other hand, some of the tags produce longer line-breaks than the others. For example, P nodes have top and bottom margins longer than those of a DIV node. Line break tags need to be further classified according to their representation in the page.

Modern web pages also contain many dynamic contents which is updated frequently or with a user interaction, without reloading the whole page. This type of content is usually invisible at the initial page load and becomes visible after an event or a AJAX call appends some additional content to the existing nodes. However, these invisible nodes are considered as invalid; therefore, they are not taken into consideration when they become valid later. The algorithm applies only on a state of a web page, in order to catch the dynamic content, it requires further execution of segmentation process.

In summary, VIPS Algorithm is a very popular and functional algorithm to segment a web page according to its both visual layout and the underlying DOM structure.

However, the algorithm needs to be improved to implement a segmentation mechanism on modern web pages.

4 VIPS Improvements and Implementation

In order to solve the limitations explained in the previous section, we have implemented an extended version of the VIPS algorithm on Accessibility Tools Framework (ACTF) on Java platform[2]. ACTF is a framework and extensible infrastructure, which enables developers to build a variety of utilities for improving the accessibility of applications and content for people with disabilities. ACTF provides a IE based browser and interfaces to control this browser, such as changing textual content of an element. Using these interfaces, DOM structure of a web page and its visual rendering properties are retrieved.

First of all, we extended the set of tags by including HTML5 tags and classifying them under tag sets which are defined in the original algorithm. Then, we have further classified line break nodes with respect to the spaces which they produce around their borders. For example, a DIV node and a P node produce different size of spaces in their visual representation. Moreover, we added some visual cues to detect separate contents in a web page. These cues include font size and font weight, background color as already exist in the original algorithm and additionally margin, padding and float attributes. These visual attributes are generally used to separate different content with titles and empty spaces between the content and create coherent content by grouping sub-blocks in the page layout. We have also implemented embedded objects such as video players and widgets in the web pages.

Another ambiguity in the original algorithm is about invisible nodes. Although there is a basic method to make a node invisible in visual representation, some designers refer different techniques which need to be explained in detail. The most general technique for invisible nodes is to set its display attribute to 'none', or visibility attribute to 'hidden'. However, in some web pages, invisible nodes are created by setting the absolute position of the node to out of page borders with either high negative left or negative top values. Another possibility for invisible nodes is that, their text-indent attribute is set to a negative value to prevent them being represented in the web page. Also, some leaf nodes without textual or graphical content has either zero width or height. These nodes are only represented in the page layout as separators between significant content.

By using the above definitions and visual cues, we traversed the nodes in DOM structure and examined the visual styles of each node. Following rules are applied on the nodes in this structure for visual block extraction:

1. Remove all the invalid nodes. Note that, valid nodes without any child must be kept, since, although they do not have a valid content, they appear in the page layout and might be used as a separator between two different topics. Also, although invalid nodes are not taken into consideration, they may be significant when dynamic content segmentation is handled and criteria for ignoring invisible nodes may be changed in future extensions.

[2] `http://www.eclipse.org/actf/`

2. If a node has only text node or invalid children, no block extracted.
3. If node has only one child,
   (a) If child is a text node or virtual text node, then no block extracted.
   (b) If child is line-break node, then same rules are applied to the child node.
4. If all the children are virtual text nodes of a node, node is put into block pool and we do not divide this rule in next turns.
5. If some of the children has full page width and some of the children are in the same horizontal axis and their total width is equal to the page width, divide this node so that, each full width node is a separate block and the nodes in the same horizontal axis are grouped into one single block.
6. If node is a table and some of its columns have different background color than the others, divide the table into the number saparate columns and construct a block for each piece.
7. If one of the child node has bigger font size than its previous siblings, divide node into two blocks. Put the nodes before the child node with bigger font size into the first block, and put the remaining nodes to the second block.
8. If the first child of the node has bigger font size than the remaining children, extract two blocks, one of which is the first child with bigger font size, and the other contains remaining children.
9. If a node contains a child whose tag is HR, BR or any of line-break nodes which has no children, then the node is divided into two as the nodes before the separator and after the separator. For each side of the separator, two new blocks are created and children nodes are put under these blocks. Note that, separator does not extract a block under the main block, it just serves to extract two blocks which other nodes are put into.
10. If a node has some nodes whose line-break spaces are different than each other, divide this node. Each node having larger line-break space will be a separate block, and other nodes next to each other create a block as a whole.
11. If node has at least one child with float value ”left” or ”right”, create three blocks. For each children,
    (a) If child is left float, put it into the first block.
    (b) If child is right float, put it into the second block.
    (c) If child is not both left and right float, put it into the third block. If first block or second block have children, create new blocks for them. Also, create a new block for the child without float.
12. If a node has a child, whose at least one of margin-top and margin-bottom values are nonzero, divide this node into two blocks. Put the sibling nodes before the node with nonzero margin into the first block and put the siblings after the node with nonzero margin into the second block.
    (a) If child has only nonzero margin-top, put the child into second block.
    (b) If child has only nonzero margin-bottom, put the child into first block.
    (c) If child has both nonzero margin-top and nonzero margin-bottom, create a third block and put it between two blocks.
13. If a node has a line-break child containing an image or object, divide node into two blocks. Put the nodes before the child with image or object into the first block, and put the remaining children into the second block.

14. If the first child of the node contains an image or image, put this child as a block, and create a new block for the remaining children.

The application precedence of the rules are the same as their definition order. In each iteration of visual block extraction mechanism, each visual block is appended in a block pool with respect to their hierarchical position in the page layout. Finally, a content structure is built by using these visual blocks in a tree structure. Our implementation concerns only on visual block extraction and content structure construction parts in the original algorithm, since visual separator detection process is not relevant with our usage area of web page segmentation.

We also committed our implementation to ACTF and it is available under the terms of the Eclipse Public License v1.0[3], so that, anyone can benefit from our contribution.

5 Evaluation

We have conducted a user evaluation to better understand how people perceive the success of our extended and implemented VIPS algorithm. In order to reach more people from different countries with different demographic properties, such as level of expertise in web design and development, age group and education, our user evaluation was conducted online which is available at `http://emine.ncc.metu.edu.tr/eval/survey/`.

Procedure

Regarding the segmentation, our online study included three main parts:

Information Sheet: First page in our study included an information sheet about the study. This page mainly included an overview of the study, and some information about the anonymity and the tasks to be completed.

Demographics: When somebody participated in our study, we collected some demographics information about them, for example their gender, experience in web design, education, age range, etc.

Visual Elements Identification: In this part, participants were shown a web page in top five levels of segmentation, and they were asked to rate these levels and also rank the levels of segmentation. In overall, the survey application was designed to repeat this step for randomly selected nine pages.

Materials

In order to group web pages with respect to their sizes and complexities, we have used the Visual Complexity Rankings and Accessibility Metrics (VICRAM) framework [15]. For a given web page, VICRAM assigns a Visual Complexity Score (VCS) which represents the complexity of the web page. We have selected top 100 web pages from Alexa and calculated their VCS values. By grouping these pages into three, we then randomly

[3] `http://www.eclipse.org/legal/epl-v10.html`

selected ten pages from each complexity group and 30 overall pages. The first group is low complexity pages with VCS value lower than three. The other group consists of medium complexity pages with a VCS value between three and seven. Final group includes high complexity pages with a VCS value above seven. However, when we were preparing the data for our evaluation, we could not take a screenshot of one of the low complexity pages because it was too dynamic therefore we have 29 pages at the end. This approach gave us a systematic method for choosing pages with different levels of complexity. Our evaluation aims to retrieve data of pages from each complexity group for each participant, so that, ranking and rating task were repeated for nine randomly selected pages, which includes three pages from each complexity group.

Research Questions

Regarding the segmentation, our study focused on investigating the following two research questions:

1. What is the perceived success of our extended segmentation algorithm? How does this differ based on the web page complexity? With the first part of the question, we aim to investigate how people view the segmentation success and in the second part we try to understand whether complexity has any effect on the perceived success of our algorithm.
2. Which level of segmentation is the most preferred? As explained above our segmentation algorithm segments a web page in different levels, therefore this question aims to understand which level would people think be appropriate level for segmentation.

Results

Our online study was released on the 7th of February 2013, and here we present the preliminary analysis of our results based of the data collected in two weeks after the survey was announced. In overall, 220 participants completed our study and 25 of them have completed at least three pages. Since our study was designed to include pages with different complexity levels, we wanted to make sure that data that we analyze come from participants who evaluated at least one page from each complexity level. Of our 25 participants, ten were female and 15 were male. Seven participants were aged between 18-24, eight of them were 25-34, eight of them were 25-54 and two of the participants were aged over 55. Five participants completed high/secondary school, two completed associate's degree, six completed bachelor's degree, seven completed master's degree and five completed doctorate. 18 participants have worked in web design and development, four of them studied this subject and three of them are interested in web design as a hobby. Seven participants describe their level of expertise in web design and development as professional, 13 as intermediate and five as novice/beginner. All of the participants use internet daily.

In overall, 259 pages and 1,295 levels have been evaluated. Among these pages and levels, 148 pages and 740 levels were considered in our evaluation as valid evaluations since their participants satisfied the minimum requirement of evaluating at least three pages from different complexity levels. Evaluated pages consist of 49 pages and 245

Table 1. Rating results

	Level 1	Level 2	Level 3	Level 4	Level 5
Low Complexity	47.2	58.4	66.4	70.45	72.22
Medium Complexity	42.4	52	66	73.6	74
High Complexity	40.8	51.6	62	68.4	76

Table 2. Ranking Results

	High Complexity					**Medium Complexity**					**Low Complexity**				
Level No	Best L.				Worst L.	Best L.				Worst L.	Best L.				Worst L.
Level 5	**38**	4	1	2	6	**31**	3	1	2	11	**17**	1	1	1	6
Level 4	5	**35**	2	4	5	6	**30**	2	7	3	5	**15**	1	2	3
Level 3	2	5	**36**	5	3	5	5	**33**	2	3	1	4	**13**	5	3
Level 2	2	2	8	**33**	5	3	4	5	**28**	8	1	3	6	**15**	1
Level 1	2	5	4	7	**32**	3	6	7	9	**23**	2	3	5	3	**13**

levels from high complexity, 48 pages and 240 levels from medium complexity, and 51 pages from low complexity.

Table 1 represents the ranking results of the participants for three complexity groups and five segmentation levels. For each value in the table, we have detected the overall view of each participant on a particular level and calculated the average result over participants' responses weighted by their population variance. The rating results indicate that, for all complexity groups, the 1st level of segmentation has the lowest success rate and 5th level of segmentation has the highest success rate.

Table 2 shows the ranking results of the participants for five segmentation levels. In this table, best level column represents the level which has been selected by the majority of participants as best level of segmentation among the first five levels in a particular web page. According to the level ranking results, the best level among first five levels of segmentation is the 5th level, in which segmentation is more detailed. The worst level of segmentation is the 1st level, where the web page is segmented into its basic segments.

Discussion

In brief, user evaluation results on levels of segmentation shows that the success rate in the 5th level is the most successful segmentation, which has a value around 74% and success rate decreases when the segmentation level decreases, down to approximately 40%. The growth in success rates, when we look at the upper levels, is the same and success rates are very similar for each level in three complexity groups. However, there is a significant difference between the success rates of 1st and 5th levels in each complexity groups.

According to the ranking results of the participants in our user evaluation shows that, when we pick the most selected level among each column in Table 2, the best level of segmentation among the first five levels in a particular web page is selected to be the

5th level, which represents the visual blocks appearing in the highest depth in the block structure. Moreover, 5th level has smaller visual blocks and more detailed segmentation when it is compared to the preceding levels. Participants also ranked the 1st level of segmentation as the worst level. This may be because the first level of segmentation does not have enough granularity and people perceived them as not so useful.

When we compare the results of rating and ranking procedures, we see that, the results are parallel in growth, so that, 5th level which is selected as the best level is also rated as the most successful level. Similarly, the 1st level which is selected as the worst level is also rated as the most unsuccessful level. However, the same rules apply on each node in every level of a web page, therefore, we expect to have similar rates for each level. This may be explained with our method of evaluation. Since we have conducted an online survey, and the pages which the participants evaluated are very limited, there is a possibility that most of the participants applied the same task of ranking the levels. Therefore, the results in the rating part of our survey may also reflect the ranking of segmentation levels rather than the success of our segmentation approach.

6 Summary and Future Work

This paper presented how we have extended VIPS algorithm for automatically segmenting web pages. The paper also presented our online user evaluation which was conducted to investigate how people perceive the success of the segmentation approach and in which granularity they prefer to see a web page segmented. Finally, the paper presented the preliminary results which show that people perceive segmentation with higher granularity as better segmentation regardless of the web page complexity. This finding is very important for being able to decide how to segment a web page automatically and display it to mobile and disabled users for better interaction experience.

Our current implementation of the extended VIPS Algorithm on the ACTF framework does not include a specific mechanism in detecting and understanding dynamic content in web pages. Many pages for example contain auto-suggest lists, dropdown menus, calendars, notification boxes, tickers, tabs, slideshows (carousels). These blocks are typically invisible at the initial load of the page and become visible after a certain event, such as the mouse comes over a menu item for dropdown menus, user types some characters in a text box for auto-suggest lists, user clicks on a node for calendars and after some events, the result of the event is displayed in the page for notifications. The invisible content is either loaded initially as set to be invisible (as in dropdown menus and calendars) or appended to an existing node in the DOM structure after an AJAX call (as in auto-suggest lists). Therefore, we need to detect the invisible content and the event which turns these content to visible. After their detection, the content should not be divided further to its sub-blocks, rather represent a single visual block as a whole.

Content expansion is also very popular in modern web pages. In this type of dynamic content, a visible content is expanded after user interaction, such as, expansion buttons and accordion menus. Another type of content expansion is page expansion, in which, new content is retrieved with an AJAX call and appended to the bottom of the page, when user scrolls down to the bottom. In page expansion, new content is not included in the initial DOM structure and has the same visual structure with existing content. These kind of dynamic content is important for users who are accessing web pages in

constraint environments. One might need to display only the new expanded material to the user, however the our extended VIPS algorithm does not explicitly focus on these expanded content and web pages need to be segmented all together. Therefore, in our future work we are planning to address this by dynamically segmenting the new content without completely reprocessing a web page.

Acknowledgements. The project is supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) with the grant number 109E251 (http://emine.ncc.metu.edu.tr/). As such the authors would like to thank to (TÜBİTAK) for their continued support.

References

1. : Efficient web browsing on small screens. In: Proceedings of the Working Conference on Advanced Visual Interfaces, AVI 2008, pp. 23–30. ACM, New York (2008)
2. Asakawa, C., Takagi, H.: Annotation-based transcoding for nonvisual web access. In: ASSETS 2000, pp. 172–179. ACM Press (2000)
3. Baluja, S.: Browsing on small screens: recasting web-page segmentation into an efficient machine learning framework. In: WWW 2006: Proceedings of the 15th International Conference on World Wide Web, pp. 33–42. ACM, New York (2006)
4. Borodin, Y., Mahmud, J., Ramakrishnan, I.V., Stent, A.: The hearsay non-visual web browser. In: Proceedings of the 2007 International Cross-disciplinary Conference on Web Accessibility (W4A 2007), pp. 128–129. ACM, New York (2007)
5. Cai, D., He, X., Li, Z., Ma, W.Y., Wen, J.R.: Hierarchical clustering of www image search results using visual, textual and link information. In: Proceedings of the 12th Annual ACM International Conference on Multimedia, MULTIMEDIA 2004, pp. 952–959. ACM, New York (2004)
6. Cai, D., He, X., Wen, J.R., Ma, W.Y.: Block-level link analysis. In: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2004, pp. 440–447. ACM, New York (2004), http://doi.acm.org/10.1145/1008992.1009068
7. Cai, D., Yu, S., Wen, J.R., Ma, W.Y.: Vips: a vision based page segmentation algorithm. Tech. Rep. MSR-TR-2003-79, Microsoft Research (2003)
8. Chen, J., Zhou, B., Shi, J., Zhang, H., Wu, Q.: Function-based object towards website adaptation. In: Proceedings of the Tenth International World Wide Web Conference. ACM, Hong Kong (2001)
9. Chen, Y., Ma, W., Zhang, H.: Detecting web page structure for adaptive viewing on small form factor devices. In: Proceedings of the Twelfth International World Wide Web Conference (2003)
10. Chen, Y., Xie, X., Ma, W.Y., Zhang, H.J.: Adapting web pages for small-screen devices. IEEE Internet Computing 9, 50–56 (2005), http://portal.acm.org/citation.cfm?id=1053547.1053593
11. Hattori, G., Hoashi, K., Matsumoto, K., Sugaya, F.: Robust web page segmentation for mobile terminal using content-distances and page layout information. In: WWW 2007: Proceedings of the 16th International Conference on World Wide Web, pp. 361–370. ACM Press, New York (2007)
12. Hwang, Y., Kim, J., Seo, E.: Structure-aware web transcoding for mobile devices. IEEE Internet Computing 7(5), 14–21 (2003)
13. Lunn, D., Harper, S., Bechhofer, S.: Identifying behavioral strategies of visually impaired users to improve access to web content. ACM Trans. Access. Comput. 3(4), 13:1–13:35 (2011), http://doi.acm.org/10.1145/1952388.1952390

14. Mahmud, J.U., Borodin, Y., Ramakrishnan, I.V.: Csurf: a context-driven non-visual web-browser. In: Proceedings of the 16th International Conference on World Wide Web, WWW 2007, pp. 31–40. ACM, New York (2007), http://doi.acm.org/10.1145/1242572.1242578
15. Michailidou, E.: ViCRAM: Visual Complexity Rankings and Accessibility Metrics. Ph.D. thesis (2010)
16. Milic-Frayling, N., Sommerer, R.: Smartview: Flexible viewing of web page contents. In: Poster Proceedings of the Eleventh International World Wide Web Conference (May 2002)
17. Song, R., Liu, H., Wen, J.R., Ma, W.Y.: Learning block importance models for web pages. In: Proceedings of the 13th International Conference on World Wide Web, WWW 2004, pp. 203–211. ACM, New York (2004), http://doi.acm.org/10.1145/988672.988700
18. Takagi, H., Asakawa, C., Fukuda, K., Maeda, J.: Site-wide annotation: Reconstructing existing pages to be accessible. In: ASSETS 2002, pp. 81–88. ACM Press (2002)
19. Whang, Y., Jung, C., Kim, J., Chung, S.: Webalchemist: A web transcoding system for mobile web access in handheld devices. In: Optoelectronic and Wireless Data Management, Processing, Storage, and Retrieval, pp. 102–109 (2001)
20. Xiang, P., Shi, Y.: Recovering semantic relations from web pages based on visual cues. In: Proceedings of the 11th International Conference on Intelligent User Interfaces, IUI 2006, pp. 342–344. ACM, New York (2006), http://doi.acm.org/10.1145/1111449.1111531
21. Xiao, X., Luo, Q., Hong, D., Fu, H.: Slicing*-tree based web page transformation for small displays. In: Proceedings of the 14th ACM International Conference on Information and Knowledge Management, CIKM 2005, pp. 303–304. ACM, New York (2005), http://doi.acm.org/10.1145/1099554.1099638
22. Xiao, Y., Tao, Y., Li, Q.: Web page adaptation for mobile device. In: Wireless Communications, Networking and Mobile Computing (2008)
23. Xiao, Y., Tao, Y., Li, W.: A dynamic web page adaptation for mobile device based on web2.0. In: Proceedings of the 2008 Advanced Software Engineering and Its Applications, pp. 119–122. IEEE Computer Society, Washington, DC (2008), http://portal.acm.org/citation.cfm?id=1487741.1488145
24. Xie, X., Miao, G., Song, R., Wen, J.R., Ma, W.Y.: Efficient browsing of web search results on mobile devices based on block importance model. In: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications, pp. 17–26. IEEE Computer Society, Washington, DC (2005), http://portal.acm.org/citation.cfm?id=1048930.1049752
25. Yang, X., Shi, Y.: Enhanced gestalt theory guided web page segmentation for mobile browsing. In: Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, vol. 03, WI-IAT 2009, pp. 46–49. IEEE Computer Society, Washington, DC (2009), http://dx.doi.org/10.1109/WI-IAT.2009.227
26. Yesilada, Y., Chuter, A., Henry, S.L.: Shared Web Experiences: Barriers Common to Mobile Device Users and People with Disabilities. W3C (2008), http://www.w3.org/WAI/mobile/experiences
27. Yesilada, Y., Harper, S., Goble, C.A., Stevens, R.: Screen readers cannot see (ontology based semantic annotation for visually impaired web travellers). In: Koch, N., Fraternali, P., Wirsing, M. (eds.) ICWE 2004. LNCS, vol. 3140, pp. 445–458. Springer, Heidelberg (2004)
28. Yin, X., Lee, W.: Using link analysis to improve layout on mobile devices. In: Proceedings of the Thirteenth International World Wide Web Conference, pp. 338–344 (2004)
29. Yin, X., Lee, W.S.: Understanding the function of web elements for mobile content delivery using random walk models. In: Special Interest Tracks and Posters of the 14th International Conference on World Wide Web, WWW 2005, pp. 1150–1151. ACM, New York (2005), http://doi.acm.org/10.1145/1062745.1062913