A Robust Vision-Based Framework for Screen Readers

Michael Cormier(✉), Robin Cohen, Richard Mann, Kamal Rahim, and Donglin Wang

Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada
{m4cormie,rcohen,mannr,krahim,d35wang}@uwaterloo.ca

Abstract. With the increasingly rich display of media on the Internet, screen reading technology that mainly considers website source code can become ineffective. We aim to present a solution that remains robust in the face of dynamically displayed web content, regardless of the underlying web framework. To do this, we consider techniques used in computer vision to determine semantic information about the web pages. We consider existing screen reading technologies to see where such techniques can help, and discuss our analytical model to show how this approach can benefit low vision users.

Keywords: Computer vision · Screen reader · Visually impaired · Sensory substitution

1 Introduction

Modern web pages are complex, and include dynamic content and content in a wide variety of languages and frameworks. While this provides a rich experience for most users, it can degrade the experience of low-vision or visually impaired users. Low vision users are users who have limited visual acuity despite using the best lenses that are available. These users typically use screen readers, which are applications that take the standard output generated from other programs, interpret it, and read it to the user in a meaningful way. Existing screen readers designed for use on web content typically use the page source code structure to determine the best way to present the content (*e.g.* ChromeVox [8] and VoiceOver [1]); the incorporation of rich media makes this much more difficult because of the complexity of the underlying page source code. The resulting output is very difficult for users to interpret.

We use methods from computer vision to help create a semantically rich landscape for users with low vision. By rendering the web page and emulating how humans inspect web pages, we hope to reduce the reliance on high verbosity screen readers, thus providing a better user experience. Verbosity settings on screen readers can be used to read when a frame begins, or when images appear on web pages, along with other formatting features of the web page. However, we acknowledge that some users enjoy a high verbosity screen reader because it can help to create a mental model of the web page in their minds. It may be valuable to think of low vision users performing web page rendering in their mind, using the output of a screen reader as their own "source code" to generate the display of the web page in their mental model. Since current screen reading technology mainly looks at web source code to determine where features lie in websites [8] [1], there is room for improvement using solutions from computer vision, such as edge detection and general feature extraction to more accurately report to users where content and formatting lies relative to one another. In contrast to other efforts, we propose an approach to the problem designed to more closely emulate the ways that sighted users perceive a web page, as a way of improving robustness.

This work proposes augmented or alternative communication for users who are attempting to view online webpages. Whether completely blind or simply visually impaired (either due to disability or to age), our methods would enable an audio conveying of webpage content for the user, as an additional option. This work therefore also constitutes sensory substitution (where visual display is prepared to instead be provided as audio output). Our proposed solution leverages techniques from computer vision in order to decide what to present as output.

This paper is divided into four sections, of which this introduction is the first. Section 2 describes our proposed approach to the problem. The advantages and disadvantages of our approach relative to other methods are discussed in Section 3. Finally, Section 4 consists of concluding remarks and proposed directions for future research.

2 Proposed Solution

We propose that a screen reader can make use of the high-level structure of a web page in much the same way that a human user does: by the appearance of the rendered page itself. While specific languages and frameworks for web design will inevitably evolve or be replaced over time, the web pages that they are used to produce are likely to remain consistent. There are common patterns in web design which are well established. Our intention is to respect these patterns; to fundamentally change these design patterns may be confusing for users. Figure 1 shows an example of one common pattern of organization for a web page. It is immediately clear that the top field is likely to contain a title and possibly links to major sections of the website, the left-side field is likely to contain links or incidental information, and the larger field occupying the central, right, and lower regions probably contains the primary contents of the page.

Not all web page layouts are this simple, but in general web pages are designed to organize information in an intuitive way that makes it as easy as possible to find the primary content, incidental content (such as contact information), and links to other sections of the website. Despite the often considerable complexity of a web page's code, a well-designed web page has a clear, simple structure.

We describe a combination of computer vision techniques and models which can be used to divide a web page into semantically meaningful regions, to classify

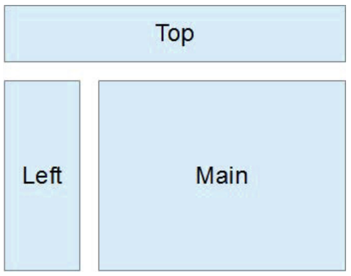


Fig. 1. Schematic example of a website structure. It is natural to assume that the main content is in the cention marked "Main," the sidebar ("Left") contains links, and the top bar contains a title and possibly some links.

these regions according to their semantic role in the page, and to read the text as the user navigates through the page. Our framework is designed to be resilient with respect to changes in the languages used to present web content and to allow visually-impaired users to navigate web sites in the same way that a sighted user would. The flowchart shown in Figure 2 shows the organization of our proposed system, with references to the sections of the text describing the major components. Likewise, Algorithm 1 describes the process of reading a page.

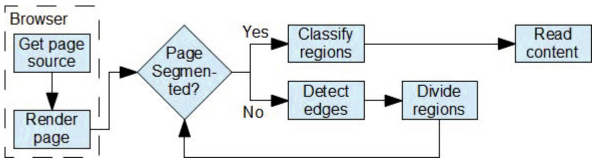


Fig. 2. Flow chart showing the high-level organization of our proposed screen reader and segmentation framework

2.1 Segmentation

Unlike previous work on segmentation for screen readers [6] [9] [12], our proposed framework uses a top-down approach to page segmentation. We believe that this is advantageous because, like a human observer seeing a page from too great a distance to see details, a top-down system can produce a reasonable large-scale segmentation even if the details are unclear or computational resources are insufficient to produce a detailed segmentation in a reasonable time. Top-down page segmentation methods have a precedent in, for example, OCR applications [3]. Our segmentation algorithm produces a hierarchical segmentation by beginning with a region corresponding to the entire page and recursively dividing regions along logical boundaries in the visual structure of each region until no such

Algorithm 1. High-level algorithm showing the process of reading a web page. The segmentation and classification functions are shown in algorithms 2 and 3, respectively.

```
1 Function ReadPage($I$)
      Input: Page image $I$
2     $v_{img} \leftarrow I$ ;
3     $v_{feat} \leftarrow$ RegionFeatures($I$) ;
4     $T_{init} \leftarrow (\{v\}, \emptyset)$ ;
5     Segmentation tree $T \leftarrow$ Segment($T_{init}$) ;
6     Classification tree $L \leftarrow$ Classify($T$) ;
7     Read regions in the order specified by the user ;
```

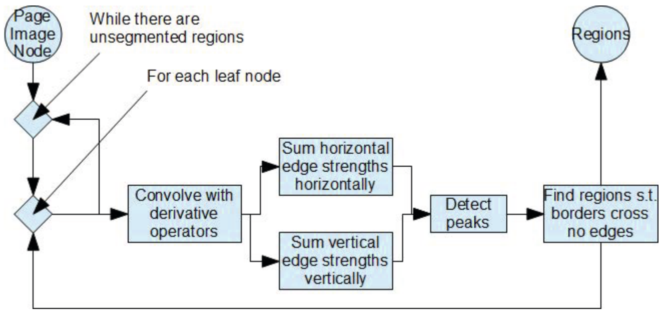
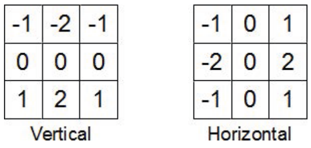

Page Image Node
While there are unsegmented regions
For each leaf node
Convolve with derivative operators
Sum horizontal edge strengths horizontally
Sum vertical edge strengths vertically
Detect peaks
Find regions s.t. borders cross no edges
Regions

Fig. 3. Flow chart showing the web page segmentation process

logical boundaries remain. Figure 3 shows a high-level description of the page segmentation algorithm.

The first step in subdividing a region is edge detection. The process of edge detection is described in detail by Castleman [2]; here we summarize the aspects of the problem applicable to our system. An edge detection algorithm treats the image as a two-dimensional discrete function I, where $I(x, y)$ is the value of the pixel at row x and column y. The simplest definition of an edge is a point at which the magnitude of the gradient is very large; that is, the image changes very quickly (since the image function is discrete, its derivatives can only be approximated). The orientation of the edge is perpendicular to the gradient, so a vertical edge has a strong horizontal gradient. Partial derivatives of the image function ($\frac{\partial I}{\partial x}$ and $\frac{\partial I}{\partial y}$) can be calculated to specifically detect edges which are approximately vertical or horizontal, respectively. One common way to calculate the gradient or derivative is to convolve the image with a discrete kernel K; different kernels perform differently in actual edge detection. Our method uses separate horizontal and vertical edge detection (using the partial derivative kernels K_h and K_v) to produce a horizontal "edge strength" function $S_h = K_v * I$ and a vertical equivalent $S_v = K_h * I$. The edge strengths, calculated by this method, represent the magnitude of partial derivatives of brightness with respect to image position. These edge strengths are converted to binary edge maps E_h and E_v using a threshold t: $E_h(x, y) = \begin{cases} 1 & S_h(x, y) \geq t \\ 0 & S_h(x, y) < t \end{cases}$; E_v is defined analogously. The threshold is a parameter of the edge detection algorithm and can be tuned for a specific application such as our screen reader system. These edge maps are likely to have edges several pixels wide; these edges can be thinned down to more accurately locate the true edges in the image; Castleman describes [2] methods for performing this operation, but in the interest of clarity we will not discuss these methods in detail here.



-1	-2	-1
0	0	0
1	2	1

Vertical

-1	0	1
-2	0	2
-1	0	1

Horizontal

Fig. 4. Sobel kernel functions for the vertical (left) and horizontal (right) partial derivatives of the image function. The vertical partial derivative is high across horizontal edges, and the horizontal partial derivative is high for vertical edges.

Since edge detection is so well studied, there are many algorithms that can be used. We use the Sobel edge detection operators [2], shown in Figure 4. Our initial results use the MATLAB Image Processing Toolbox implementation of the Sobel edge detection method, which automatically determines a threshold and thins the detected edges to a single pixel.

The features that our system intends to detect *may* correspond to long edges in the image, but may also correspond to aligned short edges. A sidebar with a distinct background colour, for example, will have long, continuous edges where the background colour changes, but while two columns of text against the same background colour should be separated, there is no continuous edge between them. Objects such as characters or images aligned to a horizontal or vertical line implicitly define a line which may be frequently interrupted by spaces between the objects. For the alignment to be visible, however, each object must have at least a short edge along the line to which they are aligned; furthermore, this edge must have a component parallel to the line of alignment. Our feature extraction system finds the estimated strength of vertical and horizontal division lines—both those formed by long edges and those formed by alignments—by summation of the edge map horizontally or vertically. The estimated vertical boundary strength in a specific column of the image is the total number of edge pixels in the column, and similarly for horizontal edges and rows in the image. It is reasonable to use only horizontal and vertical divisions between regions because the elements of a web page are likely to be vertically and horizontally organized. Once these features have been found for rows and columns in the image, isolated peaks are selected as candidate region boundaries. The edges of the image are also considered to be possible region boundaries.

The row-based and column-based boundary strengths are shown for an example image of the Facebook home page in Figure 5. It consists of the original image with aligned graphs of horizontal and vertical boundary strengths. Note that the highest boundary strengths (the peaks in the graphs above and to the side) correspond to natural divisions in the image of the web site.

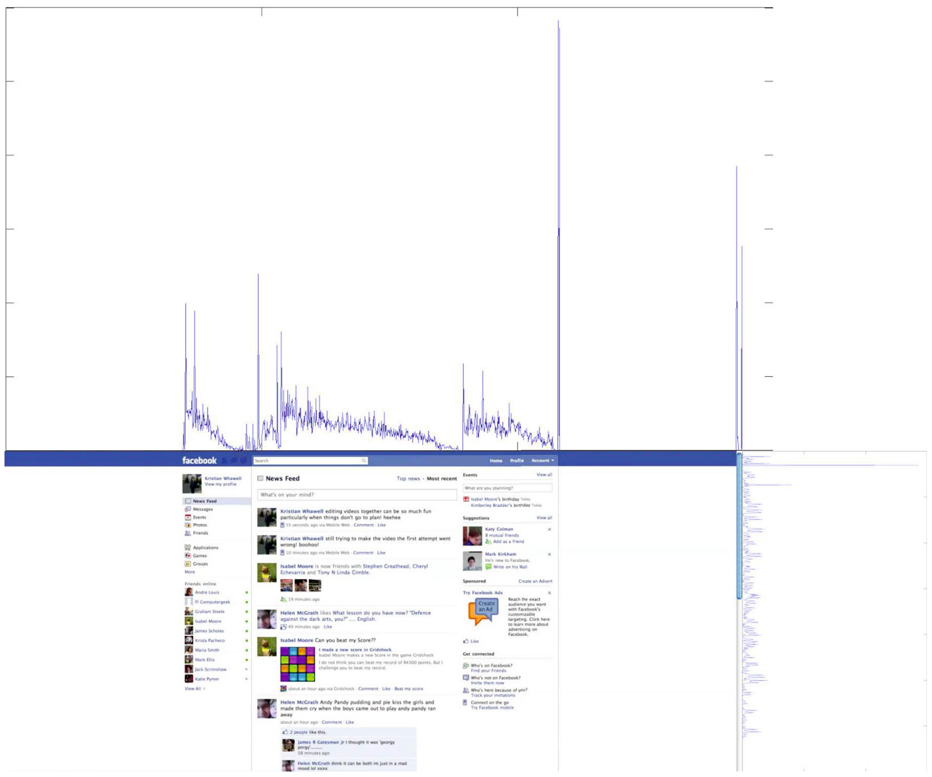


Fig. 5. Feature detection on Facebook. The top graph corresponds to vertical boundary strengths in the page image shown, and the graph on the right to horizontal boundary strengths. The vertical axis of the graph of vertical boundary strengths represents the sum of edge strengths in each column, and the horizontal axis represents position in the image and is aligned with the corresponding columns in the image of the web page. The graph of horizontal boundary strengths is similar, but rotated: replace "columns" in the previous description with "rows," and exchange the axes. Note that the strongest peaks in boundary strength are aligned with natural divisions in the page.

Once the candidate sub-region boundaries have been found, it is necessary to find the optimal set R of sub-regions that correspond to natural divisions in the current region. Each sub-region is an axis-aligned rectangle whose sides correspond to segments of the identified sub-region boundaries. Furthermore, these sub-regions must also obey the following properties:

1. No two sub-regions may overlap
2. No sub-region may include the entire current region
3. Every part of the region must be a part of some sub-region

These properties may not yield a unique segmentation. In this case, a cost function $C(R)$ can be used to guide the segmentation process. The cost function can be used to enforce a wide variety of preferences, including the number of regions, the internal properties of regions, and the relationships between regions. It may also prove useful to use domain-specific cost functions for different web pages. The requirement that the sides of the rectangle correspond to candidate region boundaries dramatically reduces the number of possible segmentations that must be searched to find an optimal solution. If valid sub-regions are found, these sub-regions are the children of the current region is the hierarchical segmentation; otherwise, the current region is a leaf node in the segmentation tree.

Figure 6 shows an example segmentation performed manually. This is an ideal segmentation, of course, but it demonstrates the objective of the segmentation algorithm. In practice, the program would segment further, but the depth of the segmentation is limited here for clarity.

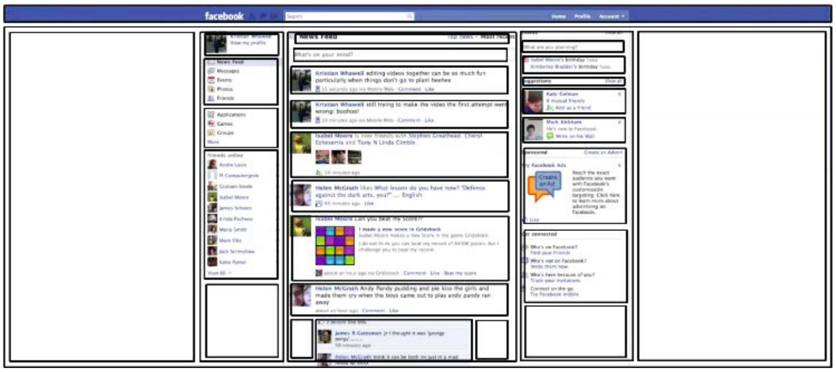


Fig. 6. Example of a manual segmentation (to a tree depth of 4) showing an ideal segmentation of a Facebook page

As in page segmentation for OCR (*e.g.* [3]), our objective is to produce a segmented page using only visual information, not code. This contrasts with typical approaches taken in web page segmentation systems (*e.g.* [4], [9]); even web page segmentation systems that use visual information typically apply bounding boxes and text properties derived from the HTML and CSS source code of the web page. Avoiding reliance on this information is intended to both ensure independence from underlying technologies (*e.g.* HTML, HTML 5, Flash, or Silverlight) and to more closely emulate the visual methods used by sighted users.

Our segmentation algorithm, representing the culmination of the methods described in this subsection, is displayed as Algorithm 2. It accepts as input an image of a rendered web page and outputs a segmentation tree of unlabelled regions in the page. The edge detection process is shown in context as part of the segmentation process.

Algorithm 2. Algorithm for producing a segmentation tree of a page image.

```
1  Function Segment($T$)
       Input: Segmentation tree $T = (V, E)$
       Output: Segmentation tree $T' = (V', E')$
2      foreach $v \in V$ such that $v$ is a leaf node do
3          $S_h \leftarrow v_{img} * K_h$ ;
4          $S_v \leftarrow v_{img} * K_v$ ;
5          $E_h(x, y) \leftarrow \begin{cases} 1 & S_h(x, y) \geq t \\ 0 & S_h(x, y) < t \end{cases}$ ;
6          $E_v(x, y) \leftarrow \begin{cases} 1 & S_v(x, y) \geq t \\ 0 & S_v(x, y) < t \end{cases}$ ;
7          $b_h(y) \leftarrow \sum_x E_h(x, y)$ ;
8          $b_v(x) \leftarrow \sum_y E_v(x, y)$ ;
9          Let $X$ represent the set of strong peaks in $b_v$ and $Y$ represent the set of
           strong peaks in $b_h$ ;
10         if $X \cup Y \neq \emptyset$ then
11             Find a set $R$ of rectangular regions with edges corresponding to
               candidate boundaries in $X$ and $Y$ such that $C(R)$ is minimized and
               all required properties are maintained ;
12             $T' \leftarrow T$ ;
13             foreach $r \in R$ do
14                 $w_{img} \leftarrow$ region of $v_{img}$ corresponding to $r$ ;
15                 $w_{feat} \leftarrow$ RegionFeatures($w_{img}$) ;
16                 $T' \leftarrow (V' \cup \{w\}, E' \cup \{(v, w)\})$ ;
17             return Segment($T'$)
18         else
19             return $T$
```

2.2 Classification

After the web page has been segmented, the regions must be classified according to their semantic role in the page in order to present the structure of the page to the user in a comprehensible way. The classification system should have three key properties. First, the classifications of regions should account for their context in the page, not just the appearance of the specific region. Second, the structure of the classification should reflect the hierarchical structure of the segmented regions. Finally, the classification system should be flexible and adaptable to account for unusual web page designs. We propose to use a classification system based on a hidden Markov tree (HMT), a type of probabilistic graphical model originally developed for wavelet-domain signal processing [5]. Much like a hidden Markov model (HMM), an HMT contains two types of node: observations and hidden states. Whereas hidden states in an HMM have a simple sequence-based dependency structure, however, the hidden states in an HMT have a tree-based dependency structure. This model is depicted as a Bayesian network in Figure 7.

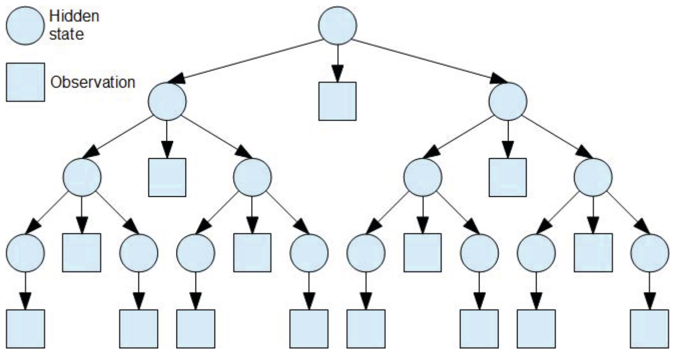


Fig. 7. A hidden Markov tree (HMT) with hidden states (circles) and observations (squares)

The tree structure of the hierarchical segmentation corresponds to the structure of the HMT. The hidden states in the HMT represent the labels of the regions in the segmentation, and the observations in the HMT represent feature vectors describing the regions.

Region labels should correspond to structures perceived by a human user—such as headers and sidebars—rather than plain structural descriptions such as "vertically-oriented container." This is critical because it is these classifications that will be used to generate the descriptions of the page layout that the user will use to navigate the page.

The choice of features used as observations in the HMT is critical to the success of the classification system. We propose to use a combination of visual features (colour and texture, for example) and position features. The visual features allow the system to distinguish between, for example, text and images. Position features allow the classification system to account for the broader context in which a region occurs.

The combination of visual features, position features, and the label of the parent region allow the classification system to respect broad patterns (which is similar to the rules used in Project ABBA [9]). Consider, for example, the case of a pattern consisting of a header, a left sidebar, and a body (as shown in Figure 1). There will be, *a priori*, a probability of $\frac{1}{3}$ that each child node will have the given pattern, but the observations of positions and sizes will clarify the matter: the left sidebar is on the left, the header at the top, and the body occupies a large percentage of the page. The observations, and the requirement that child regions be non-overlapping and jointly cover an entire parent region, capture dependencies among sibling regions, allowing the use of a simple HMT model for classification.

The classifier would be initially trained from a hand-labelled data set consisting of segmented web pages, since the web page source code would not reliably provide clear evidence of the semantic roles of regions. This training process would only need to be performed once, as a final step in development, and the resulting classifier would be used as the default system for all users. The use of learning would also make it possible to update the classifier in response to user feedback about classification errors. This in turn would allow the system to adapt to the browsing habits of individual users and the designs of the web sites that each user typically visits.

The algorithm for classifying the regions on the page (the second function in our high level algorithm (Algorithm 1)) is displayed as Algorithm 3. It accepts a hierarchical segmentation (in a tree format) and produces a tree of region labels.

Algorithm 3. Algorithm for classifying regions according to their semantic roles.

```
1 Function Classify($T$)
      Input: Segmentation tree $T = (V, E)$
      Output: Classification tree $T' = (V', E')$
2     Let $H = (S, O, F)$ be an HMT, where $S$ is the set of states, $O$ is the set of
      observations, and $F$ is the set of edges ;
3     For each node $v_i \in V$, let $s_i \in S$ represent the label of the region
      corresponding to $v_i$ ;
4     For each node $v_i \in V$, let $o_i \in O$ represent the observations (extracted
      features) for $v_i$ ;
5     $F \leftarrow \{(s_i, s_j) : (v_i, v_j) \in E\} \cup \{(s_k, o_k) : s_k \in S\}$ ;
6     Calculate the most likely set $S'$ of states given the observations $O$ and
      learned conditional probabilities ;
7     $T' \leftarrow (S', E)$ return $T'$
```

2.3 Reading

Robustness is the key concern in reading the page contents. When reading a block that has been identified as a text-containing block, the system will attempt to detect text in the image. When text has been found, the system attempts to match the text in the image of the block with text embedded in the page source code. A successful match allows the system to simply read the text found in the page source. If no match is found—for example, if the text is embedded in an image—the system can fall back to existing OCR techniques (see Figure 8).

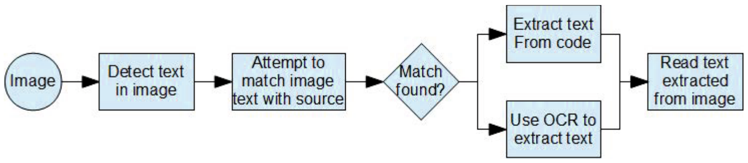


Fig. 8. Flow chart showing the output process

In use, the system would provide a brief description of the page to the user, who would select the region to read. If the region can be read immediately, the system does so; otherwise, the system describes the structure of the region to the user, who selects a sub-region to read. This continues until a simple text region is reached, which is then read to the user. Navigation of the page is in terms of the classified regions rather than in terms of the code hierarchy.

2.4 Next Steps

The choice of cost functions in segmentation is a significant outstanding question. A good cost function should reflect beliefs about the typical structure of a segmentation tree. It could be used to place approximate limits on the number of divisions at each step, encourage internal similarity within regions, or encourage regions to be of similar sizes. Empirical testing will be required to find an effective cost function.

The system presented here uses two stages—segmentation and classification—to produce the tree used to navigate the structure of the web page. It is worth considering if the labels determined at the classification stage could be used to modify the tree structure. This may allow the system to recover from a poor initial segmentation, increasing robustness and making the details of the segmentation cost function less important.

Practical issues such as implementation and interface design also merit further consideration. Integration with the web browser as an extension is a natural way to implement a screen reader system, although other options may be viable. The interface should be carefully designed to present information about page layout quickly (so that the user is not slowed down significantly) but clearly (so that the user can apply the information effectively). A method for selecting regions to read should also be quick and simple to use.

3 Discussion

There are various screen readers available today. Many of the existing screen readers such as ChromeVox [8] and VoiceOver [1] are based on the source code of web pages, i.e. HTML code, to retrieve the content of web pages. However, such a method has drawbacks. It is difficult to identify the main content solely based on source code. Some text under a "<div>" tag could be a meaningful sentence that a reader is interested in, or a table header that serves no real purpose. As a result, such screen readers could potentially read out texts that are useless to users. Moreover, as web languages evolve quickly, different websites might use different technologies for the development of their web pages. Methods such as source code analysis would require constant updates to comply with new standards, which is not scalable. Also, many web pages today support dynamically loaded content using scripting languages such as Javascript. Content generated by these scripts may not be found in the HTML source code which limits the usability of such methods.

Our solution differs from the above mentioned method in two areas. First of all, our solution is mainly based on the layout of a web page to identify where the main content is. This is similar to how humans interpret a web page when he or she sees it. Users see a web page as merely an image with static objects. They can quickly identify where useful information is by decomposing the image into segments. This is the aim of our solution: by using techniques such as edge detection, our solution divides a web page into regions, providing a clear indication of the semantic structure of the page. This helps label regions on the page. For example, once the main content area is identified, the user can tell the program to read out only the content from the main section using, for instance, a keyboard shortcut. This has significant advantages over existing screen readers. Many existing screen readers do not discriminate contents of different regions and they often read out the entire page. From a user's perspective, this is not appropriate because many web pages have the same header across multiple pages. Reading out these contents every time a user visits a web page is redundant. Furthermore, labelling the regions on pages can help users navigate through the website. For example once a left side bar is identified, a user can tell the program to read out the left bar so that he can interact with the links on the page.

Another area of difference our solution has over existing methods is that we use a visual model for segmenting the page and classifying regions. This more closely resembles the way in which a sighted user perceives a web page. Since web pages are often designed specifically for sighted users, we believe that this is a significant advantage over existing screen reader software.

In ChromeVox, a user needs to navigate the code hierarchy to the desired content in order to reach the information he or she is interested in [8] [11]. On the contrary, our solution does not search for content based on a code hierarchy, but rather makes use of computer vision techniques to jump directly to the content inside the code. Another common screen reader is Apple's VoiceOver. This screen reader is natively built into Mac OSX, and iOS devices [1]. This screen reader performs well on native applications, in part because Apple can couple the development of their products with the VoiceOver application, ensuring the output is up to specification. However, on the web, where content is not controlled by Apple, VoiceOver may not perform as well. Basing the OCR output on source code also has the challenge of trying to offer a coherent depiction for users. An example of this problem can be seen in the demonstration performed by Youtube user "kwhawell" [10]. This demonstration of Apple's VoiceOver system compares the fairly good performance on Facebook's simple mobile web site to the poor performance on the sophisticated desktop version. On the desktop version, VoiceOver jumps between regions which appear visually distinct and it is difficult to reach the status update form field. Our vision-based system, on the other hand, would be able to reflect the visual organization of the page in the structure it generates for navigation.

Project ABBA [12] has produced screen reader software that uses the visual properties of web site regions. Our system differs in its approach in the following areas:

- Our system is designed specifically to handle content presented in formats other than HTML and CSS gracefully, rather than to rely on the page source code.
- Our system is designed to reproduce the experience of sighted uses insofar as that is possible, rather than using a specialized navigation model.
- Our system uses a learned probabilistic graphical model to classify blocks in the page rather than manually-defined rules; this allows online learning to adapt the system to individual users' browsing habits

These differences make our approach complementary to existing work.

The primary disadvantage of our solution is speed. Vision algorithms can be computationally intensive [7], and a complete web page corresponds to a very large image. This disadvantage is mitigated by the increasing speed of modern hardware, however, so this need not be a fatal flaw. Web pages that rely on animation or the use of large numbers of complex images may confuse the segmentation algorithm, although web pages where images and animations dominate text content are less likely to be useful to visually-impaired users in any case.

It is important to note that our design was inspired by certain existing research on segmentation and classification for information retrieval and optical character recognition, though there as well some fundamental differences. Page segmentation methods in OCR—as in the system proposed by Cesarini *et al.* [3]—are designed to divide the page into regions suitable as input for the OCR algorithm, not input suitable for a human navigating the page. These segmentation algorithms do not need to label regions. Information retrieval methods (such as the system described by Chen, Zhong, and Cook [4] based on a generalized hidden Markov model), in contrast, produce very fined-grained segmentations and classifications which isolate specific fields and values (such as phone numbers or prices) for an automated system. The goal of web page segmentation for human users is distinct; unlike segmentation for OCR, regions must be labelled, and the desired labelled regions are coarser and at a higher level than for IR methods.

4 Conclusion

We believe that the visual structure of a rendered web page is the most natural source of information about the large-scale organization of information in the web page, since the rendered page is designed to be easy to interpret visually. We believe that our proposed method can provide a significant advantage over screen readers that do not use visual information. Furthermore, our approach differs in many respects from existing work to develop a screen reader which uses visual information. Because our system has minimal reliance on the semantics of the page source code, it can readily handle new web technologies without fundamental changes to the system itself.

Next steps with this research include proceeding with a validation. We envisage a user study, for example comparing the proposed method with a direct translation of webpages (e.g. VoiceOver [1]), as one component. We are also interested in more detailed specification of the execution times of the proposed method and its inherent computational complexity.

Future work may also include extending the method to user interfaces other than web pages (such as office software interfaces) and providing finer-grained control over the types of information the system uses. Other valuable directions for future work include providing more opportunities for personalized solutions for users and investigating in greater depth the specific needs that arise in social networking environments, such as Facebook, with a large number of messages. As the users who are interested in our proposed framework may have varying levels of sight, the output that we produce may possibly be complementary to existing displays (rather than replacing them entirely). It would be interesting to enable users to specify certain subsets of content that they would like to serve as the input to our algorithms. In addition, if users can indicate which elements they are not particularly interested in, the output that is delivered from our system can be reduced. User preferences may also motivate an extension to our system, where the visual representation driving our production of output is first of all cast in terms that focus best on what the user is most interested in knowing about. As for additional exploration of the context of social networks, we can imagine that social networks may evolve over time in order to be more welcoming of visually impaired participants; we may be able to propose certain adjustments to the gathering of information streams in these networks, to then proceed from these representations to more effective output for our users.

References

1. Apple: Apple - accessibility - OS x - VoiceOver. http://www.apple.com/accessibility/osx/voiceover/ (accessed March 16, 2014)
2. Castleman, K.R.: Digital Image Processing, chap. 18. Prentice Hall (1996)
3. Cesarini, F., Gori, M., Marinai, S., Soda, G.: Structured document segmentation and representation by the modified x-y tree. In: Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR 1999, pp. 563–566, September 1999
4. Chen, J., Zhong, P., Cook, T.: Detecting web content function using generalized hidden markov model. In: 5th International Conference on Machine Learning and Applications. ICMLA 2006, pp. 279–284, December 2006

5. Crouse, M., Baraniuk, R., Nowak, R.: Hidden markov models for wavelet-based signal processing. In: Conference Record of the Thirtieth Asilomar Conference on Signals, Systems and Computers, pp. 1029–1035, vol. 2, November 1996
6. Fayzrahmanov, R.R., Göbel, M.C., Holzinger, W., Krüpl, B., Baumgartner, R.: A unified ontology-based web page model for improving accessibility. In: Proceedings of the 19th International Conference on World Wide Web. WWW 2010, pp. 1087–1088. ACM, New York (2010). http://doi.acm.org/10.1145/1772690.1772817
7. Forsyth, D.A., Ponce, J.: Computer vision: a modern approach. Prentice Hall Professional Technical Reference
8. Google: ChromeVox. http://www.chromevox.com/ (accessed March 16, 2014)
9. Krüpl-Sypien, B., Fayzrakhmanov, R.R., Holzinger, W., Panzenböck, M., Baumgartner, R.: A versatile model for web page representation, information extraction and content re-packaging. In: Proceedings of the 11th ACM Symposium on Document Engineering. DocEng 2011, pp. 129–138. ACM, New York (2011). http://doi.acm.org/10.1145/2034691.2034721
10. "kwhawell": introducing a screen reader what it is how it works (2010). https://www.youtube.com/watch?v=mklnYoge7pk (accessed June 16, 2014)
11. Raman, T.V., Chen, C.L., Mazzoni, D., Shearer, R., Gharpure, C., DeBoer, J., Tseng, D.: Chromevox a screen reader built using web technology. Google technical report (2012). http://google-axs-chrome.googlecode.com/svn-history/r165/trunk/developer/chromevox-overview-2012/paper.pdf
12. TUWIEN Database and Artificial Intelligence Group: TUWIEN Project ABBA: Web Accessibility. http://www.dbai.tuwien.ac.at/proj/ABBA/ (accessed June 4, 2014)