Computer Vision on Web Pages: A Study of Man-Made Images

by

Michael James Cormier

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2018

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	John Tsotsos Professor, Dept. of Electrical Engineering and Computer Science York University
Supervisor(s):	Robin Cohen Professor, Cheriton School of Computer Science, University of Waterloo Richard Mann Associate Professor, Cheriton School of Computer Science, University of Waterloo
Internal Member:	Daniel Vogel Associate Professor, Cheriton School of Computer Science, University of Waterloo
Internal Member:	Kate Larson Professor, Cheriton School of Computer Science, University of Waterloo
Internal-External Member:	Christopher Small Professor, Dept. of Statistics and Actuarial Sciences, University of Waterloo

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis is focused on the development of computer vision techniques for parsing web pages using an image of the rendered page as evidence, and on understanding this under-explored class of images from the perspective of computer vision. This project is divided into two tracks—applied and theoretical—which complement each other. Our practical motivation is the application of improved web page parsing to assistive technology, such as screenreaders for visually impaired users or the ability to declutter the presentation of a web page for those with cognitive deficit. From a more theoretical standpoint, images of rendered web pages have interesting properties from a computer vision perspective; in particular, low-level assumptions can be made in this domain, but the most important cues are often subtle and can be highly non-local. The parsing system developed in this thesis is a principled Bayesian segmentation-classification pipeline, using innovative techniques to produce valuable results in this challenging domain. The thesis includes both implementation and evaluation solutions.

Segmentation of a web page is the problem of dividing it into semantically significant, visually coherent regions. We use a hierarchical segmentation method based on the detection of semantically significant lines (possibly broken lines) which divide regions. The Bayesian design allows sophisticated probability models to be applied to the segmentation process, and our method produces segmentation trees that achieve good performance on a variety of measures.

Classification, for our purposes, is identifying the semantic role of regions in the segmentation tree of a page. We achieve promising results with a Bayesian classification algorithm based on the novel use of a hidden Markov tree model, in which the structure of the model is adapted to reflect the structure of the segmentation tree. This allows the algorithm to make effective use of the context in which regions appear as well as the features of each individual region.

The methods used to evaluate our page parsing system include qualitative and quantitative evaluation of algorithm performance (using manually-prepared ground truth data) as well as a user study of an assistive interface based on our page segmentation algorithm. We also performed a separate user study to investigate users' perceptions of web page organization and to generate ground truth segmentations, leading to important insights about consistency.

Taken as a whole, this thesis presents innovative work in computer vision which contributes both to addressing the problem of web accessibility and to the understanding of semantic cues in images.

Acknowledgements

I would like to thank my supervisors, Prof. Robin Cohen and Prof. Richard Mann. They have provided valuable advice and assistance from the beginning of the degree program. They have given me the freedom to pursue my research interests, while supporting my work with their expertise in research and in the academic process. More specifically, I would like to thank Prof. Mann for his emphasis on mathematical foundations and on clarity of presentation, and Prof. Cohen for her assistance in smoothing the process at the level of logistics and organization, and with connecting with other researchers.

I would like to thank my committee members, Prof. Dan Vogel, Prof. Kate Larson, Prof. Christopher Small, and Prof. John Tsotsos for their work in reviewing the thesis and their advice in preparing the final version. The clarity of the thesis in particular has been greatly improved through their advice.

I would like to thank our frequent collaborator Prof. Karyn Moffatt of the School of Information at McGill University for her advice with respect to the application domain of assistive technology and assistance in establishing contacts in this research community. Several research assistants have also done valuable work in the process of this thesis: Anthony Chan, Mengfei Liu, Michael Parrott, and Shangshang "Daniel" Zheng. My friends and colleagues John Doucette, Dan Recoskie, and Alan Tsang have provided helpful insights and companionship during my studies. Many members of the University of Waterloo community have provided valuable support "behind the scenes": Helen Jardine, Wendy Rush, Margaret Towell, and Loc Do have been of great help in administrative matters; Mike Gore and Lawrence Folland have provided responsive and skilled technical support for my research.

My studies and research have been funded by a number of organizations and scholarships. I would like to specifically mention NSERC funding, through the NSERC PGS-D program and my supervisors' Discovery grants, and the David R. Cheriton Graduate Scholarship.

On a more personal note, I would like to thank my parents, Rose Cormier and Prof. James Cormier, for their personal support over the course of my doctoral program and, indeed, throughout my education. None of my academic accomplishments would have been possible without them.

Finally, I would like to acknowledge Marcus Aurelius, whose *Meditations* has provided valuable perspective on the difficulties and frustrations inevitable in the course of any large and complex undertaking.

Dedication

I would like to dedicate this thesis to my parents, Rose and James Cormier, and my late grandfather Gerry Gartland.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The focus of this thesis is the development of a computer vision system for parsing web pages based solely on the visual appearance of the rendered page[1]. For the purposes of this research, "parsing" refers to the process of determining the structure of the page *content*, not the page *implementation*. This is quite distinct from the problem of parsing, for example, the HTML source code of the page. Parsing the page content involves identifying key regions in the page such as an article blurb or a header. This can be done using evidence from the implementation of the page—and, as will be seen, this is a commonly-used method—but it is a separate problem from parsing in the sense used in programming languages. Our parsing method is based on a segmentation-classification pipeline. The page is first segmented into visually coherent, semantically significant regions forming a segmentation tree. These regions are then classified to identify their roles in the page, resulting in a hierarchical parse tree of the page, including both the locations and roles of the various regions that make up the page. Figure 1.1 shows this segmentation-classification pipeline: the source code is rendered by the browser to produce an image of the page; the segmentation algorithm produces a segmentation tree from the rendered page, and the classification algorithm takes the rendered page and segmentation tree to produce a parse tree, which is finally used for practical purposes by some other program such as a screen reader.

There are a wide range of images that can be considered "man-made" in some sense. In one sense, any photograph can be considered man-made, since it is technologically produced, but this is not a meaningful distinction for computer vision applications, which

[1] Our published papers point to earlier versions of some of the work presented here, namely [32], [35], [31] and [34] for Chapter 3 and [33] for Chapter 4.

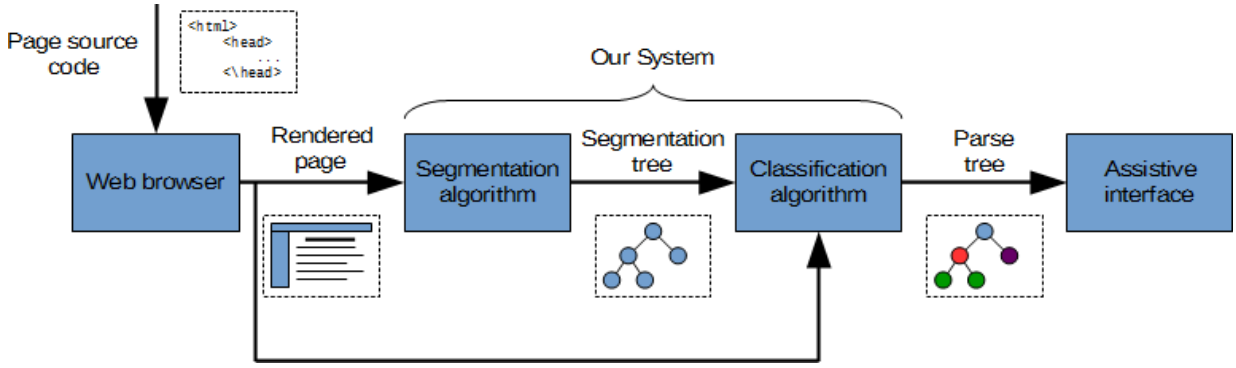


Figure 1.1: Diagram showing the segmentation-classification pipeline described in this thesis in context, as a component of an assistive interface.

inherently require images captured by some form of man-made sensor. In the sense in which we use the term here, man-made images are images in which the visual structure of the image was produced intentionally, through a design process, rather than arising naturally from projection of a scene without human intervention (although natural images may form a component of a man-made image). Such images could include paintings and drawings, as well as infographics, document layouts, and advertisements (when they are not a simple photograph, which would be considered a natural image). We are particularly interested in web pages as representatives of a sub-class of man-made images in which the visual layout is intended to reflect the semantic structure of information, rather than the physical structure of a scene.

A simple example of manual parsing of a web page can be seen with an examination of a Google search results page. The page, shown in Figure 1.2, shows a web page structure with many regions made distinct using a variety of cues. Figure 1.3 shows how the page can be parsed, at a high level, by a user. The regions shown here are can be parsed further into smaller regions. The main results section, for example (Figure 1.3c) can be parsed into seven regions, each consisting of a link to, and excerpt from, one of the pages returned as a search result. Each of these regions could be parsed further (based on colour information) into the link, the URL, and the excerpt. Clearly, even a relatively simple page with a minimalist design can produce a large and complex parse tree. Note also that there are clear semantic differences between, for example, a search-result blurb and the page header. For many applications it is reasonable to assume that these regions should be distinguished based on their semantic role and treated differently, while two result blurbs should be treated similarly.

This research has two complementary tracks: one applied, and one theoretical. The

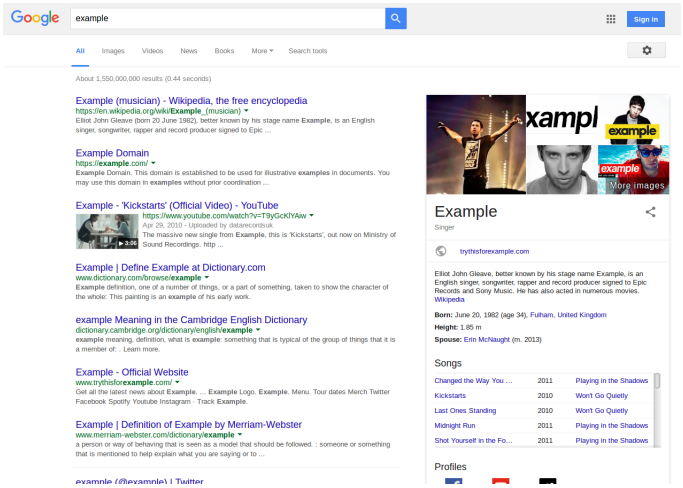
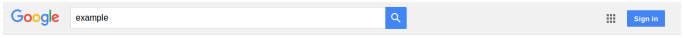
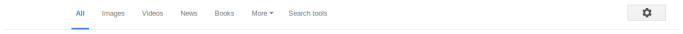


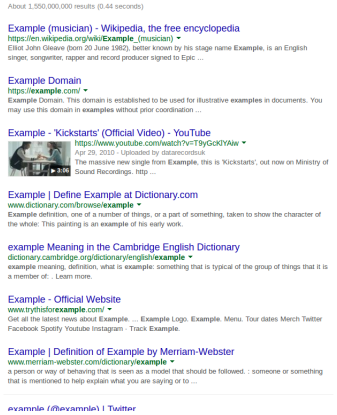
Figure 1.2: Example of a relatively simple web page.



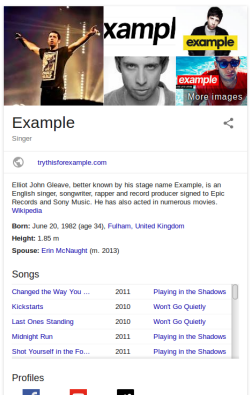
(a) Upper header region, distinguished by a difference in background colour



(b) Lower header region, distinguished by an outline



(c) Main results, distinguished by whitespace



(d) Sidebar, distinguished by an outline

Figure 1.3: Manual parse of the largest regions in the page shown in Figure 1.2

applied track is focused on the applications of visual parsing of web pages to assistive technology, although the proposed algorithms are potentially useful in other areas such as information retrieval and web design. The theoretical track consists of the development of computer vision techniques for understanding this under-explored class of image using principled Bayesian[2] methods. These two approaches support each other: theoretical study of web pages as a class of images helps to inform the development of parsing algorithms for this domain, while the application domain suggests desiderata for the output of these parsing algorithms and experimental results from practical tests raise questions for further theoretical research. Similarly, both tracks lead to important contributions. Assistive technology is an important area of research, particularly considering the needs of an aging population [62], and our work in this area contributes to providing more effective solutions for web accessibility, as we will demonstrate through user studies and offline testing. More theoretically, our exploration of web pages as a domain for computer vision algorithms and our development of web page parsing algorithms with a strong theoretical grounding provides valuable insights into computer vision, with broader implications for other classes of image. Ultimately this thesis is best seen as research into computer vision, motivated and guided by socially important applications to assistive technology.

Our initial motivation in studying visual parsing of web pages was to investigate the possibility of improving web accessibility. Users with a wide range of physical, sensory, and cognitive disabilities may struggle to use web pages as they are normally presented. Visually impaired users, of course, have difficulty seeing the page, and individuals with cognitive, reading, or learning disabilities can find it difficult to process cluttered and content-heavy pages. Assistive interfaces, such as screen readers for visually impaired users (which convert the depiction of a web page into an audio transcript), can help to address these challenges, but the complexity of modern webpages creates significant challenges for users of assistive interfaces. Screen reader users can, for example, find it difficult and laborious to reach desired content in a page. While guidelines exist for making webpages more accessible, research has shown that these best practices are often ignored [60, 131]

Existing assistive interfaces primarily use the source code or the Document Object Model (DOM) tree[3] to determine the semantic structure of the page in order to create an appropriate alternative presentation of the page. There are frameworks to assist in the automatic interpretation of the source code structure, such as WAI-ARIA labels (see Section 4.3); unfortunately, these are not consistently or reliably used by web designers.

[2]For the purposes of this thesis, we use the term "Bayesian" in the sense that we make extensive use of Bayes' Theorem in relating *a priori* and *a posteriori* estimates of probability. This is different from the sense of systematic use of integration over parameters.

[3]An intermediate representation of the page produced by the browser

Furthermore, implementation details can change over time as new frameworks and language versions are introduced (requiring corresponding updates to the parsing algorithm used by the assistive interface), and some objects such as infographics are "black boxes" with no internal structure visible through source code or the DOM tree. These factors lead to problematic implementation-dependence for conventional methods for parsing the semantic structure of web pages. A vision-based method using the visual appearance of the page avoids implementation dependence by entirely avoiding the implementation of the page, focusing instead on the rendered page produced by the browser.

More fundamentally, the designer or designers of a web page create the visual appearance of the page to clearly and simply convey the semantic structure of the page contents to "typical" users (as the designers imagine them). The source code of the page, in contrast, is simply designed to produce the correct appearance and functionality. We therefore argue that the visual appearance is in some sense "closer" to the intended semantic structure of the page. Of course, the visual appearance of the page is designed to be clear to humans rather than to machines; as a result, sophisticated computer vision methods must be developed in order to leverage the visual appearance of the page in a page parsing algorithm. It is the development of these methods, and their implications for computer vision research, that form the core contributions of this thesis.

Our research contributes to the study of parsing a broader class of man-made images in which, as in web pages, visual cues (*e.g.*, layout, alignment, and background colour) are used to convey semantic relationships between regions. Web pages have particularly useful features as exemplars of this class of image, since the rendering process results in "well-behaved" low-level properties, and there are vast numbers of readily-available examples. As an example, while many images in this class can be modelled by rectangular regions, only rendered images such as web pages guarantee that these regions are aligned to the image axes; it is therefore possible to ignore the initial stage of estimating the transformation between the coordinate systems of the image and of the document. Of course, a document could in principle be aligned sufficiently carefully with a scanning system and made sufficiently flat to make the image and document coordinate systems identical, but this process would be very labour-intensive and may not be possible for rare or valuable documents (for example, it may not be possible to flatten the pages of a book enough to avoid distortion near the spine without damaging the binding). As a result, using the rendered image of a web page allows the study of the higher-level properties without relying on preprocessing steps which may be time-consuming or introduce errors.

Images of web pages have a number of interesting properties from the perspective of computer vision research. Although they are man-made images rather than natural images, they are designed for human perception, not for the strengths of computer vision

algorithms. Some useful assumptions about the properties of these images can be made. Because web pages are rendered images, the "camera" position is not arbitrary; the image axes are significant without any additional alignment step. Similarly, there is no need to account for blurring of the image or perspective projection outside of natural images embedded in the page, and it is often sensible to treat embedded natural images (as opposed to infographics or other images that behave as complex document components) as atomic objects in the page. At a higher level, it is reasonable to assume that regions in the web page have axis-parallel rectangular boundaries due to common design conventions and the convenience of implementing such regions in the CSS box model [66]. Despite these assumptions, however, parsing a web page still requires the interpretation of complex cues such as long-range alignments, complex layouts with large numbers of regions and potentially deep hierarchies, and the presence of regions of many different types (including text, natural images, line drawings, and many others). As a result, images of web pages can be seen as an intermediate class between natural images—with all the complexity of natural scenes—and the toy problems often used for testing specific aspects of a computer vision algorithm in a controlled setting. As a domain, images of web pages are restricted but not trivial, making them an excellent domain for the development of sophisticated algorithms grounded in a principled analysis of the properties of a class of images, with the potential for extending these algorithms to natural images. In fact, due to the ubiquity of web pages, their persistence across many types of devices, and the fact that they are used by nearly everyone, this under-studied class of images is especially valuable to examine.

Image segmentation is the process of dividing a page into semantically meaningful regions. Our proposed segmentation algorithms take a hierarchical, top-down approach, recursively dividing regions until no further divisions are possible. This produces a segmentation tree, in which small regions are grouped together as the children of the larger regions that contain them. Both segmentation algorithms presented in this thesis use as evidence the presence or absence of lines in the page.

A Bayesian framework is used to determine the likelihood that semantically significant lines exist, and these estimates are used to find divisions in each page given the page image and a prior probability distribution over segmentation structures. Each region is divided according to the maximum *a posteriori* likelihood segmentation of the region. Our approach uses a Bayesian segmentation architecture, in which successive edge detection, line detection, and segmentation steps are performed in a Bayesian setting. This principled Bayesian approach is particularly advantageous for this class of images: assumptions about segmentation structure can be made because of the restricted domain. These assumptions are sufficient to narrow the search space in order to make the search for a high-quality segmentation tractable. Within this framework, we are able to perform experiments to

test the effects of different prior probability distributions and probability models. Also of interest is that by using explicitly calculated probabilities based on suitable prior probability distributions rather than an unnormalized energy function we are able to determine thresholds based on probabilistic reasoning (*e.g.*, accepting divisions that are more likely to be valid than not). The fact that taking our algorithms' probability estimates as genuine probabilities produces plausible segmentations, is evidence for the validity, at least in broad terms, of these models.

Segmentation is valuable in that it tells us *where* the semantically significant regions of the page are, but for many applications it is also important to know *what* these regions are. Classification of regions in the segmentation tree is therefore an important process. Our classification algorithm takes as input the segmentation tree and the image of the page. From this input the segmentation algorithm creates feature vectors describing each region in the segmentation tree. It is reasonable to expect, based on knowledge of web page structure, that regions of similar appearance will have different roles depending on their context in the page. Consider, for example, a line drawing. If the line drawing appears in the header of the page, it is likely to represent the logo of the web site or the organization which owns it; it if appears in the main article, it is more likely to represent some sort of diagram or illustration. It is therefore important to use a rich and informative representation of the context of each region in the page, including the roles of its parent and child regions. It is also desirable to use a Bayesian approach to classification, for the same reasons that this is a desirable approach for segmentation. Both of these objectives are accomplished through the innovative use of a probabilistic graphical model called a hidden Markov tree (HMT). The structure of the HMT is adapted to each page to reflect the structure of the segmentation tree; regions are classified by simultaneously finding the MAP (maximum *a posteriori*) labels for all regions.

Returning to the applied track, it is worth considering in more detail the types of assistive interfaces a visual parsing system could support. Although a high-quality parsing algorithm could in principle support a wide range of assistive features, in our work we focus on the application of our system to three types: screen readers, magnifying interfaces, and decluttering interfaces.

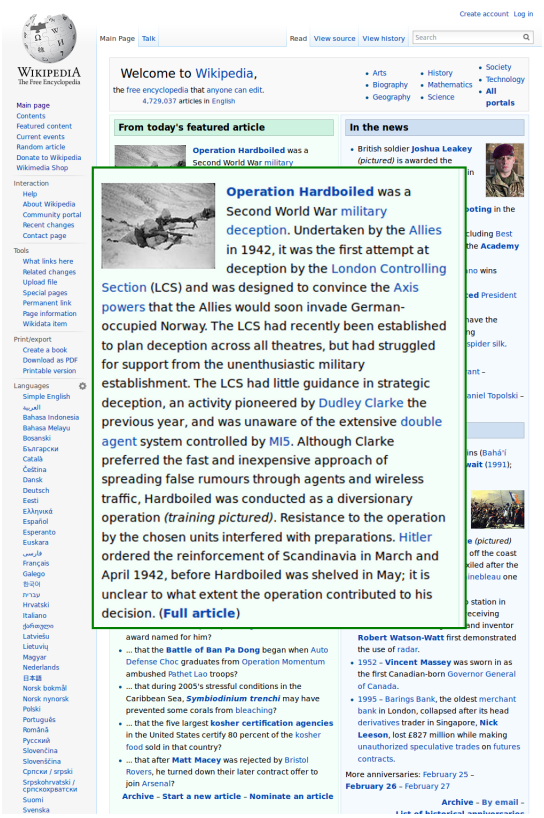
A major component of browsing the web is locating content areas within each page and assessing that content against current information needs. While sighted web users can quickly locate main content areas on the page and skim them for relevance, users with severe visual impairment relying on screen readers are often forced to process content sequentially. Because this can be a slow process, screen readers typically provide mechanisms for skipping forward. For example, users can often jump ahead by using structural markers (such as <h> tags) to move between content areas [19]. Unfortunately, such approaches rely heavily

on the quality of the underlying code. The code structure also tends to be complex, resulting in page segmentations that consist of many small segments of text. As a result, screen reader navigation is much more difficult than browsing a page with full access to two-dimensional visual cues. An analogy with computer games may illuminate the difficulty of navigating between regions in a page with a screen reader. Compare the problem of navigating through a maze-like level with a clear overhead view, to navigating through the same level in a text adventure game such as "Adventure" [37]; it is clearly much easier in the former case. A web page implementation that does not reflect the semantic structure of the page contents, or has misleading cues due to constructions used in contexts for which they were not intended, can leave users of screen readers and other assistive technology "in a maze of twisty little passages, all alike" [37].

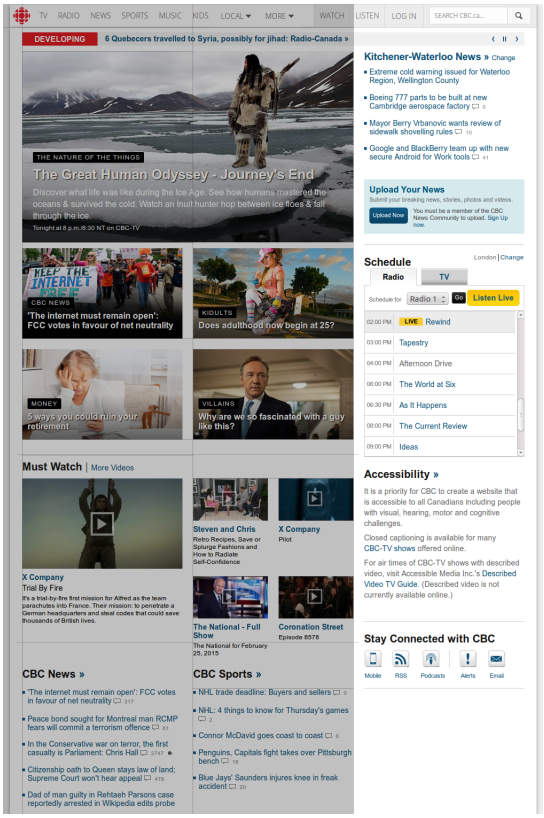
Not all users who are visually impaired require a screen reader program. For some, it may be sufficient to simply present regions at a larger scale. Individuals with certain types of cognitive impairment can also benefit from larger text [107]. Simply enlarging the entire page may obscure the large-scale organization of the page, however. Similarly, accessibility features such as Windows Magnifier or the Picture-In-Picture Zoom on Mac OS X, which use a fixed magnification boundary, may not allow the user to read an entire visually coherent region at once. We propose an interface which selects an entire visually-coherent region to enlarge as an overlay, as shown in Figure 1.4a. Allowing the user to reposition the overlay would assist in maintaining context in the page, as the users could uncover nearby regions at will to select the next region they would like to enlarge. Another avenue to explore would be to combine this approach with a "fisheye" distortion around the enlarged areas (as discussed by Furnas [50]) so that the enlarged area does not obscure any of the content; this, however, may prove confusing or distracting for some users.

Figure 1.4b shows another possible adaptation that could be achieved with our segmentation, this time designed for users with cognitive impairments or attention deficits. In this design mock up, all segments of the webpage are greyed out except the one currently selected. This approach could help users to maintain control over their area of focus. Other methods, such as blurring, could be used instead of or in combination with greying out non-focus regions, depending on what method is found to be most effective for users in general or on the preferences of individual users. Both magnification and decluttering (based on greying out non-focus regions) are available in the prototype assistive interface tested in Chapter 5.

The remainder of this thesis is organized as follows. In Chapter 2, we discuss background information that is helpful in understanding the motivation for this research in more detail. This includes the state of assistive technology on the web and the reasons to believe that a vision-based approach holds promise. Chapters 3, 4, 5, and 6 form the core



(a) A page region found by our segmentation method shown as an enlarged overlay on the original page. Note that the full width of the page remains visible if the user is able to move the enlarged section around, thus maintaining a view of the overall page structure while enlarging the text the user is reading at the time.



(b) Example page (cropped vertically) with all but a focus region greyed out to reduce distraction. The size of the focus region can be varied by selecting a node at a higher or lower level in the segmentation tree or by specifying the desired node size, in accordance with the preferences of individual users.

Figure 1.4

of the thesis; each focuses on one major aspect of the thesis research. Chapter 3 describes the segmentation algorithms developed and the experiments used to evaluate them, including both qualitative and quantitative results. Chapter 4 describes our work on region classification, using a hidden Markov tree model. Chapter 5 describes a study of an assistive interface based on our segmentation algorithm, and includes discussion of a user study and offline tests of the complete system. Chapter 6 describes our research into human perception of the semantic structures of web pages and methods for eliciting ground truth segmentations (methods which are in many respects applicable to other domains as well). Having presented the research performed in the course of the thesis, we move on to discuss related work in more detail in Chapter 7, and contrast existing work with our approach to demonstrate its novelty. Finally, in Chapter 8, we summarize the key contributions of our work and discuss the avenues for future research that it has revealed.

Chapter 2

Background

In this chapter we discuss the context which motivates our work in the visual parsing of web pages. The purpose of this chapter is to provide necessary background information prior to the discussion of the research performed in the course of this thesis; as such, the focus is not on the detailed mechanics of related work, but rather on the broader issues involved. Detailed comparisons of approaches to web page parsing and assistive technology are presented in Chapter 7. We begin with a brief description of some important features of the web page rendering process that influence the work described in this thesis. Having laid this groundwork we discuss, in general terms, how assistive technology works on the web today and what challenges remain to be addressed in this area. In Section 2.3, we discuss why, given the considerations described in previous sections, vision-based web page parsing is necessary. The discussion of the motivation for using vision-based parsing is continued in Section 2.4, which discusses more theoretical reasons for the study of images of web pages from the perspective of computer vision. Section 2.5 provides background information related to image segmentation and region classification, as these are the two key steps in our segmentation-classification pipeline.

2.1 Web Page Rendering

Although we do not propose to discuss in detail the entire page rendering process, as most of the complexities of the process are beyond the scope of this work, it is useful to provide some background information about those aspects of the process that directly affect page parsing tools.

Modern web pages are complex, and are implemented through a variety of languages and frameworks. HTML provides a skeleton and some of the appearance of the page; other aspects of the appearance are controlled using CSS (Cascading Style-Sheets). Many pages also have dynamic content implemented in Javascript and other scripting languages (*e.g.*, infinite scrolling and cycling headlines). The process of rendering a web page is, therefore, the process of generating the visual appearance of the page as displayed from this complex source code. Web development is a complex area, and both standards and practices evolve quickly. The complexity and rapid change in the way web pages are implemented is a major challenge for assistive interfaces that rely on interpretation of implementation details to present the page appropriately, and is one factor in motivating our vision-based approach.

Because of the complexity of the implementation of a modern web page, a common representation is needed. This common representation is provided by the Document Object Model (DOM) tree. The DOM is an object-oriented logical model, defined in a W3C recommendation [108]. The description of a document in the DOM takes the form of a tree or forest, reflecting a hierarchical logical organization of the document structure [108]. A DOM representation of a page is created by web browsers as an internal representation of the page during the rendering process; this is generally known as the DOM tree, to reflect its hierarchical structure. Scripts and extensions can read and alter the DOM tree. Dynamic content can be implemented through scripts modifying the DOM tree; an assistive interface can be implemented by reading the DOM tree and either modifying it to provide an alternative presentation on the screen or producing an entirely new presentation such as a audio stream. As will be shown, this is a very common approach, but one which has important drawbacks.

Most web pages are based on axis-parallel, rectangular objects. The CSS box model, for example, models each region as a rectangular content region plus surrounding padding, border, and margin areas [66]. Broadly speaking, this is a reasonable structural assumption for typical content. Most photographs, for example, are rectangular; paragraphs are approximately rectangular; articles are made up of stacks of paragraphs; and so on. There are occasional exceptions, but in most cases an axis-parallel rectangle is a good model for units of content. As a result of the strong tendency toward axis-parallel rectangles in web pages, we are able to make useful assumptions in our vision-based methods.

2.2 Assistive Technology and the Web

The complexity of modern webpages creates substantial accessibility barriers for users with a wide range of physical, sensory, and cognitive disabilities. For example, screen reader

users can find it difficult and laborious to reach desired content in a page, while individuals with cognitive, reading, or learning disabilities can find it difficult to process cluttered and content-heavy pages. While guidelines exist for making webpages more accessible, research has shown that these best practices are often ignored [60, 131]. In response to these challenges, researchers have explored a number of approaches to segmenting pages and reducing page complexity.

2.2.1 Screen Readers

A screen reader is a program that converts content originally displayed visually on a screen into an audio stream; these programs are used by users with visual impairments to access content which would otherwise be unavailable to them. Although screen readers are an important means of accessing online content for users with severe visual impairments, there are significant challenges in using them effectively. A sighted user with access to the two-dimensional rendered page can quickly shift focus from one region of the page to another with a mere eye movement. A screen reader user, on the other hand, must navigate between regions by explicitly redirecting the focus point of the screen reader program. This must also be done with limited or no direct access to the layout cues used to express the structure of the page.

Many popular screen readers such as Google's ChromeVox [53], Apple's VoiceOver [7], and NVDA [97] are based on the source code or the DOM tree of the page to interpret and read out the content of the page. These methods can be readily integrated into a browser, but they suffer a number of important drawbacks. It is often difficult to distinguish between important and unimportant content solely based on source code. A block of text under a "`<div>`" tag could be a highlighted warning or summary of interest to all readers interested in the content of the page; alternatively, it could be a boilerplate copyright statement that is irrelevant to most users. This can easily result in "cluttered" output that wastes users' time on irrelevant sections of the page. Furthermore, as web languages and especially web frameworks evolve quickly, different websites might use different technologies for the development of their web pages, even when the visual appearances of the rendered pages are very similar. Methods such as source code analysis would require constant updates to comply with new standards, which is not scalable. Many modern web pages load content dynamically using scripting languages such as Javascript; the resulting changes in the page represent an additional challenge for any screen reader.

Various accessibility frameworks have been proposed to help address these issues. WAI-ARIA (or, more commonly, "ARIA") role labels are intended to allow web developers to

include semantic labels for elements within the source code, for the use of screenreaders and other assistive interfaces [129]. ARIA role labels are discussed in detail in Section 4.3. These labels, however, are often not included in the implementation of the web page, and cannot in any case be applied to elements within images or Flash objects [19]. As with many other aspects of accessible web design, ARIA labels are often ignored or used inconsistently in web site development. Because they are a server-side technology and must be incorporated into the development process, supporting frameworks and standards such as ARIA labels imposes a burden on developers to support a relatively small proportion of the user base, and is therefore likely to be abandoned to cut development costs.

One method used in popular screen readers to facilitate understanding of the layout of a web page is an adjustable level of verbosity. When configured for high verbosity, screen readers can explicitly state when a frame begins, when images appear on web pages, and other formatting information. Some users enjoy a high verbosity screen reader because it can help to create a mental model of the web page in their minds; adjustable verbosity is intended to allow users to receive the level of detail that is most useful to them. 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 own mental model. It is important to note, however, that we do not claim that this model is the same as the visual appearance of the page as rendered by a web browser; it is possible that these users imagine a different set of cues for semantic structure. Goble *et al.* discuss how web navigation differs between visually impaired and sighted users, and include a discussion of how this relates to physical navigation [52]. Current screen reader technology detects semantically significant formatting and layout cues primarily based on the DOM tree and source code of the page [53] [7] [97]; it is reasonable to believe that there is room for improvement using solutions from computer vision to more accurately describe these cues, especially when the implementation of the page does not follow best practices or is otherwise unclear.

Screen readers have been the most prominent area for research into advanced assistive technology for the web, although the principles developed in this domain have great potential to be used in other types of assistive interfaces. Asakawa and Takagi [11] developed a system that transcodes existing Web pages with manual annotations to add structure that can be used for reordering visually fragmented grouping according to performance. Because manually annotating pages is very labour intensive, follow-on work focused on automatically annotating similarly structured pages across multi-page websites [121]. Yesilada et al. [136] use a structured ontology to identify visual segments in a web page and re-engineer pages to facilitate navigation using a screen reader. Mahmud, Borodin, and Ramakrishnan [89] introduced CSurf, a system that uses web page partitioning techniques

from natural language processing and machine learning to capture the context of a link to enable the screen reader to start reading the next page starting from the most relevant section. HearSay [18] extends this approach to additionally facilitate movement between the segments of a page.

2.2.2 Other Assistive Interfaces

Where work has targeted cognitive accessibility on the web, it has often focused more on the viability of potential supports than on the technical challenges associated with implementing them. Lee explored the general acceptability of a number of scaffolds for managing visual complexity, including visually highlighting core content, reading content aloud, and rendering page segments as buttons [85]. While these features received positive support from two groups of informants (people working with seniors and working with individuals who have developmental disabilities), the paper did not address how these supports could be implemented at large. IBM developed an enhanced open-source browser that provided a wide range of accessibility transformations, including—through manipulations of the web page's DOM tree—clutter-reducing features such as adjustment of font-size and line-spacing, stopping animations, hiding backgrounds, and reformatting pages for a single-column layout [58]. While these enhancements can help users to manage page complexity, they do not directly address the problem of excess content. Kurniawan et al. compared five different personalized web page presentations [83]. While the tools included in their evaluation were quite different from the approach we consider here, two important lessons emerge: first, personalization places extra demands on the user which must be balanced against the benefit gained from the personalization; and second, approaches that remove or filter content create a trade-off between simplification and faithful representation. Thus, approaches that can ease reading and navigation while preserving layout and content will generally be preferable.

2.3 Practical Motivation for Visual Parsing

Our vision-based approach to the problem of parsing web pages is motivated by the important practical advantages of this type of approach. In this section we discuss the first-principles reasons to believe that a vision-based approach is not only viable but preferable to other methods.

It is important to consider the purposes of the visual appearance of a web page, its DOM tree, and its source code. The designer of the page designs the layout and other visual

cues—colour, alignment, whitespace, borders, and so on—to convey the semantic structure of the page to a "typical user" as imagined by the designer. Furthermore, these cues are intended to be perceived by viewing the page on a screen, not as a list of properties. The primary role of the source code is to produce the desired appearance (as well as functional features of the page such as comment sections and dynamic content). Through frameworks such as ARIA labels, it is possible to support additional functions (including accessibility) in the source code; although useful, however, these additional features are often neglected due to complexity, cost, development time, or simple lack of awareness. Even clarity of structure and standards compliance may be sacrificed in order to meet a schedule or budget. The DOM tree of the page is a standardized representation of the page produced by the browser during the rendering process; it does not contain anything that is not specified (implicitly or explicitly) by the source code and any subsequent modifications by scripts. As a result of these considerations, we can conclude that an image of the page is in some sense "closer" to the semantic structure of the page than the implementation mechanisms such as the source code or the DOM tree.

Vision-based parsing of web pages has the important advantage of implementation independence. Even if some content is not generated through the standard web page rendering process, a vision-based parser can understand its structure. A notable case where this occurs for part of the contents of a page is images that contain structured content. While it is often possible to obtain the same layout and presentation using HTML and CSS (especially with HTML5), images are still very commonly used for graphs, infographics, and other types of structured content. These images are "black boxes" from the perspective of a method that relies on HTML and CSS source code or the DOM tree of the page, but an image of the rendered page shows the contents of these objects as a user would perceive them. Vision-based parsing therefore allows seamless parsing of embedded content that is inaccessible in the DOM tree, including images and Flash objects. In some cases the implementation structure is visible but difficult to interpret. In modern web pages, many regions are defined using `<DIV>` and `<SPAN>` tags, differentiated only by CSS class names, which may vary between pages for similar content.

Implementation independence also applies to changes in language versions. Page parsing systems that rely on the source code or DOM tree often have rules specific to HTML tags or other features, which may change over time, as in the transition from HTML 4.0.1 to HTML5, which introduced many features including the versatile `<CANVAS>` element, which allows complex drawing operations. Parsing algorithms designed for one version obviously cannot make use of new features introduced after they were designed without an update, which imposes a significant maintenance burden. The significance of this is best shown through an example.

VIPS [22], proposed by Cai *et al.* in 2003, is one of the most prominent DOM tree-based segmentation algorithms and has inspired a considerable number of descendants. It is based on a set of rules that determine when and where the algorithm should divide a region. The following tags and DOM node types are given special treatment in these heuristics:

- Inline text nodes
- `<TABLE>`, intended to represent a table but often used for layout, especially at the time
- `<TR>`, representing a table row
- `<TD>`, representing a table cell
- `<P>`, representing a paragraph

Additionally, HTML tags are used to determine the degree of coherence for a given block in the segmentation, which plays an important role in guiding the segmentation process. New tags introduced in HTML5 would, at best, simply use the default set of rules and a default degree of coherence value; to include the new tags would require a extending the algorithm to recognize them and determining the appropriate way to process them—a significant amount of work. Akpinar and Yesilada have provided an updated version of VIPS that can handle new tags [2, 3]; the maintenance burden is not necessarily insurmountable, but it can be significant, especially given the smaller community of developers working on accessibility technology than on the core functionality of browsers like Google's Chrome. The problem of smaller user and developer communities is discussed in [119], which touches on users' impressions of the issue.

A purely image-based algorithm, on the other hand, has no interaction with the page source code; only the appearance of the rendered web page affects its performance, so it is not sensitive to implementation details. It would, therefore, require much less (if any) modification to support new technology. As browsers are updated to properly render new versions of HTML, CSS, or other implementation languages, the assistive interface could make use of this work by simply capturing and analyzing the rendered page, thus reducing maintenance requirements for the corresponding assistive interfaces.

2.4 Web Pages as a Computer Vision Dataset

In addition to the specific pragmatic reasons for studying vision-based parsing of web pages, there are more general reasons to study this problem, and it has implications for computer vision more generally.

Images of web pages are designed to take advantage of human perceptual cues. These cues evolved in the context of natural scenes, but a web page looks very different from a natural scene. Studying the similarities and differences between these classes of images may offer important insights into what makes these cues fundamental, and what cues may be culturally determined rather than innate. Furthermore, images of web pages are man-made and designed to be simple to interpret, but are designed to take advantage of human perceptual cues rather than the strengths of computer vision algorithms. We suggest, therefore, that such images form a useful intermediate class between toy problems (where the image class is designed or selected to play to the strengths of computer vision) and natural images (where few simplifying assumptions can be made).

2.5 Computer Vision Segmentation and Classification

In this section we briefly describe some established computer vision solutions for image segmentation and region classification, in an effort to situate our approach with respect to the field. Chapters 3 and 4 present our solutions for segmentation and classification, respectively, in detail.

2.5.1 Edge Detection

Edge detection is one of the key low-level problems in computer vision. Many higher-level algorithms assume the existence of some form of edge detection or include an edge detection method suitable to their needs in the algorithm. Snakes, active contours, and level set methods are a good example of such algorithms, which perform higher-level detection of semantic contours and segmentation based on low-level edge detection.

Perhaps the most famous paper in edge detection is Canny's 1986 "A Computational Approach to Edge Detection" [23], which systematically defined criteria for optimal filters for detection of arbitrary edge types, focusing on the development of a step-edge detector. Canny defined three principal criteria for a good edge detection algorithm:

- **Good detection** requires that the edge detection method have a low probability of false negatives and a low probability of false positives; a perfect edge detector would report an edge if and only if an edge exists.
- **Good localization** requires that the detected edges correspond well in location to the true location of the edge; a perfect edge detector would place each detection exactly on the center of the edge in the image.
- **Single detection** requires that edges in the image do not produce multiple edge detector responses; a perfect edge detector would detect each edge in the image exactly once.

Adaptive thresholds for edge detection are critical to the performance of many algorithms. Heath *et al.* [64], for example, demonstrated that selecting parameters for edge detection algorithms on a per-image basis provides a significant improvement in performance, as rated by human judges, for several common edge detectors.

The locally adaptive thresholding technique described by Rakesh *et al.* [106] uses statistical methods. This covariance-based approach, like our method for estimating the probability that an edge is locally significant, uses the local distribution of estimated gradients to determine whether an edge is sufficiently strong relative to its neighbourhood to be detected as an edge. There are a number of differences in details, such as the use of a soft kernel-based neighbourhood by Rakesh *et al.*, but the principal differences are that our method uses Bayesian statistics to produce an estimate of the *probability* that an edge is locally significant, rather than producing a score which is then thresholded, and that our method uses a nonparametric estimate of the complete distribution of edge strengths rather than using summary statistics such as the covariance matrix. We believe that the use of a nonparametric representation of the distribution of edge strengths is advantageous, especially in our domain where the distribution is very far from a Gaussian as shown in Figure 2.1.

Statistical models of image properties can also be used in edge detection (*e.g.* [38] and [43]). Our system uses a different approach to statistical edge detection. Rather than comparing a pair of windows in the image, we compare a single measurement of preliminary edge strength to a pair of adjacent neighbourhoods, using a nonparametric representation of the distributions of preliminary edge strengths in these regions. Our approach is intended specifically to find the edges of the texture elements (such as characters in a paragraph of text) at the edges of texture regions; in the domain of rendered web pages, it is reasonable to assume (due to the rendering process) that these edge segments will be very precisely aligned.

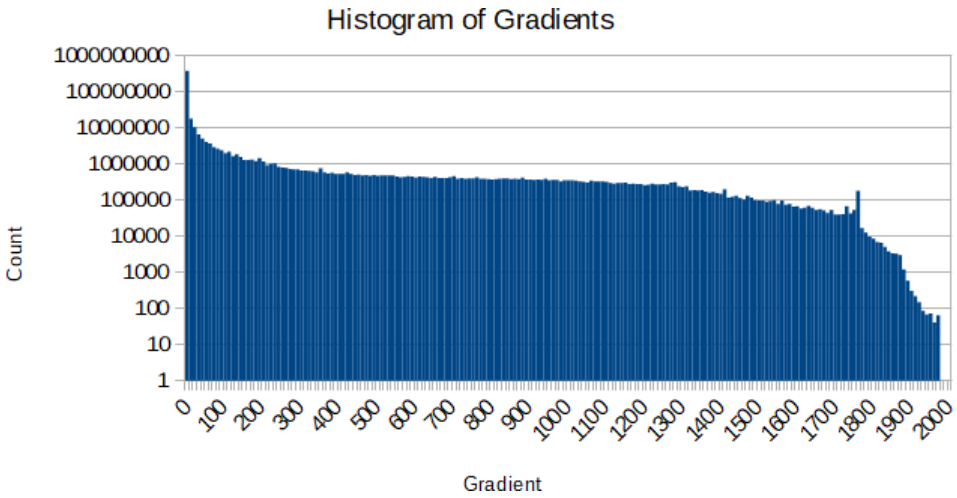


Figure 2.1: Histogram of gradients in a dataset of 100 web pages (gathered from the Alexa list of top web pages in Canada). Note that the vertical axis is shown on a logarithmic scale; most pixels in the dataset have zero gradient, and a linear scale would obscure the structure of the rest of the distribution.

One example of edge detection by learning the appearance of edges is the system proposed by Konishi *et al.* [78]. In this approach, images were processed with filter banks to produce vectors of filter responses for each pixel. Like our approach to edge detection, Konishi *et al.* used a nonparametric model to estimate edge probabilities. In the case of our algorithm, however, this distribution is derived solely from the neighbourhood of the pixel in question. This is an important difference. Our algorithm is also based on *a priori* assumptions about the properties of the domain, while the algorithm described by Konishi *et al.* focuses entirely on learning from the data.

2.5.2 Image Segmentation

The problem of image segmentation is, put simply, the problem of partitioning an image into semantically meaningful regions. It is a complex and ill-posed problem. In many cases, multiple different segmentations can be considered "correct". A remarkable range of approaches have been taken to image segmentation

Our page segmentation method uses edges as the primary evidence for segmenting a page. This approach to image segmentation has a long history in computer vision. It is an intuitively appealing approach; it is reasonable to expect that semantically significant

regions are likely to have a different appearance from their surroundings, and this will often create strong edges along the border between regions. The borders of regions are one way of defining a segmentation of the image, so edges are an appealing form of evidence to examine.

A classic example of edge-based segmentation is the snake model proposed by Kass *et al.* in 1988 [74]. Snakes are dynamic splines, moving under physically-inspired forces to nearby semantically significant contours. Active contour methods are related to our approach in that they produce segmentations supported by edges while imposing structural constraints on region shapes. The structural assumptions made by our method are, however, very different from those made by active contours. Because of the web page rendering process, we assume that the relevant edges are straight and axis-parallel, while most active contour methods penalize but allow strong curvature if it is supported by evidence in the image. Our segmentation system is also inherently hierarchical, and incorporates a prior probability distribution over segmentation trees.

Graph cut clustering has historically been a prominent technique in image segmentation. Broadly speaking, graph cut clustering uses a graph representation of data in which nodes in the graph represent data points (generally pixels or superpixels in the case of image segmentation), and the weighted edges between nodes represent affinity between the data points. The algorithm produces from this initial graph two or more disjoint clusters of data points by cutting edges. The edges to be cut are chosen to optimize an objective function based on their weights, possibly under other problem-specific hard or soft constraints. Different objective functions and constraints produce different algorithms (*e.g.* Wu and Leahy's minimum cut algorithm [133] and Shi and Malik's normalized cut algorithm [117]).

Methods based on segmenting an image into foreground and background regions can be augmented with a shape prior for the foreground region, implemented by adding a shape term to the objective function. One interesting example of this is the star shape prior proposed by Veksler [126]. Strong results from this work demonstrate the utility of a shape prior in segmenting natural images. Our page segmentation algorithm uses a shape prior in which regions are required to be rectangular; in our case, the use of the shape prior reflects the known properties of the domain and allows easier optimization of our probabilistic objective function. Veksler's approach also allows the sensitivity of the affinity of edges to adapt to local changes in the prevalence of edges. Although implemented differently, the objective is similar to our use of the local distributions of edge detector responses to estimate the probability that a possible edge is locally significant (see Sections 3.2.2 and 3.3.1).

Another common method for Bayesian image segmentation is the use of Markov random

field (MRF) models, which express spatially localized probabilistic relationships between regions. Related to the MRF model is the conditional random field (CRF) model [84]. The objective of the CRF model is to explicitly model the joint *conditional* probability distribution of a set of labels given a set of observations. In the case of image segmentation, the labels are generally region assignments and the image data constitutes the observations.

Although MRF and CRF techniques are principled, Bayesian algorithms, they differ significantly from the approach taken in our algorithms. Our approach explicitly finds the probability that specific cues truly exist in specific places in the image (*e.g.*, estimating the probability that an edge is significant with respect to its local neighbourhood, or that a semantically significant line exists between two points). The problem is divided into a series of stages, with higher-level cues building upon lower-level cues to produce the segmentation. Most segmentation algorithms based on random fields (*e.g.*, [63]) estimate the probability of an assignment of segmentation labels to the pixels (or superpixels) in the image. Our approach of more gradual assembly fits well with the edge-based approach (although edge-based potentials can certainly be implemented in random field models) and with the intended prior probability distribution over segmentation trees, and is an intuitively appealing approach to web page segmentation.

2.5.3 Region Classification

Region classification, in the sense used in this thesis, is the process of associating semantic labels with regions in a segmentation of an image. This approach is not as common in computer vision as simultaneous segmentation and classification (described in Section 7.1), or object detection. Object detection finds objects of a specific class or classes in an image, but only provides a coarse localization through a bounding box. Although not the largest area of research, there are computer vision algorithms that are designed specifically to perform region classification. It is also worth noting that region classification can be addressed using general classification methods from machine learning, with only the features used to describe regions being derived from computer vision.

Our region classification approach represents a novel classification method well-suited to the problem of classifying regions embedded in a hierarchical segmentation tree: introducing a hidden Markov tree (HMT), as described in Chapter 4. The use of an HMT with a global structure corresponding to the global structure of the segmentation tree provides a sophisticated and principled means of accounting for context without requiring context features. It allows joint inference of the maximum *a posteriori* assignment of labels over all regions in the image, while the guaranteed tree structure of the model allows efficient inference.

While our HMT-based approach to region classification is novel, hidden Markov trees have been used in other ways in computer vision, especially when working in the wavelet domain. Wavelets are a class of alternative representations of images; the wavelet transform is broadly analogous to the Fourier transform in that it is a change of basis to an alternative representation which is more appropriate for some types of analysis. The HMT of the form proposed by Romberg *et al.* in [111] has its statistical model explored in depth and many aspects of the statistics of wavelets in natural images are examined.

While the structure of the HMT shown in this research appears superficially similar to the structure of our own segmentation tree, there are important differences. Aside from its size, the structure of the wavelet HMT is fixed, not only locally but globally. It is not based upon a segmentation which adapts to the structure of the image, but rather on fixed fields in the image. Rather than representing a high-level object or region class, the hidden states in the wavelet HMT represent components of a mixture model from which the wavelet coefficients are drawn. We do not claim that the wavelet HMT is inferior to our own HMT classification model; it is a powerful model of the statistical structure of an image with many applications. Our HMT for classification simply solves a different problem, and so while it interesting to compare the two in terms of structure, the models do not compete with each other.

In [110], Romberg *et al.* used an HMT to classify textures in an image. The system simultaneously segments the page into regions of consistent texture and classifies the textures. This classification algorithm works at the level of textures and uses a fixed tree structure. Our classification algorithm uses much higher-level classes, and adapts its structure to an existing segmentation tree. Despite the superficial similarities, our system is very different in its assumptions and its approach to the classification problem.

Chapter 3

Web Page Segmentation

In this chapter we discuss the segmentation algorithms developed in the course of this thesis. These algorithms form the foundation of the other experiments described in this thesis; segmentation is the first stage in the segmentation-classification pipeline we examine here. Section 3.1 describes the motivation for developing these segmentation algorithms with respect to both applications and computer vision research more broadly. Section 3.2 describes the first algorithm developed, and Section 3.3 describes an improved version developed based on the results of the first. In Section 3.4, we describe possible applications of the segmentation algorithm in more detail. Note that in Chapter 7 we discuss how our segmentation methods relate to computer vision in general.

3.1 Motivation

From an application perspective, our research into vision-based page segmentation is motivated by assistive technology for the web. The objective of vision-based page segmentation, in the sense used here, is to determine the hierarchical structure of a web page layout using visual cues, without reference to the implementation of the web page. Our intention is for this system to serve as a back-end system, supporting front-end systems that reformat the web page for presentation to the user. Many such front-end systems, such as screen readers, exist today. Existing back-end systems for depicting web pages may use visual cues, but extract them from visual attributes defined in the code. As code-based analysis is brittle, we want to instead leverage the image of the rendered page. We believe that this approach has three principal advantages:

1. It does not depend on the quality or implementation language of the underlying code (provided that the browser's rendering engine can handle it).

2. It allows for semantically significant divisions within images, Flash objects, and other entities that are treated monolithically in HTML or CSS code.

3. Perhaps most importantly, it analyses the web page's structure using as evidence the page designer's view of the page (the appearance of the rendered web page)[1].

Essentially, the advantage of an image-based analysis is that it depends not on the details of *how* the visual structure of the page is produced, but rather on *what* the visual appearance *is*. It uses exactly the information seen by users who do not require assistive technology to make the same type of inference about the structure of the page contents. In this chapter we present a robust, extensible Bayesian framework, grounded in a formal model of web page appearance, for performing image-based segmentation of a web page, together with a comparison between the results of such an analysis and more traditional code-based techniques. As we shall see, assistive technology systems that rely on source code-based segmentation algorithms face challenges when there are, for example, images or Flash objects in the page. These algorithms would only be able to treat these objects atomically, and would be unable to detect their internal structure. As a result, users who require distracting content to be suppressed would not be able to select only parts of these objects for display.

Concrete applications of a page segmentation system include magnification and decluttering. Magnification can be useful for users with impaired vision (much like large-print books are). Decluttering can be helpful for users with cognitive impairments such as attention deficits [5], or those who simply prefer less-cluttered pages and interfaces. The latter class includes elderly users [92], who may also benefit from magnification. These interfaces are discussed in greater detail in Section 3.4. Segmentation can also serve as the first stage in a parsing system based on a segmentation-classification pipeline. Classification can considerably extend the range of uses of the parsing system, and is discussed in Chapter 4.

From the perspective of computer vision research in general, web page segmentation is an interesting problem for several reasons. First, the domain of web pages allows the use of valuable assumptions about region structure. With these assumptions the process of

[1]This is similar to the view of Saund and Moran, who argued (in the context of an image editor) that is desirable to consider visual objects that reflect what users are likely to perceive and want to manipulate [115].

optimizing over segmentations can be simplified, and more sophisticated techniques that would be too slow with more general region types can be evaluated. Second, the cues used to indicate region structure in web pages can be subtle: long-range alignments between objects, faint edges, and edges in highly textured regions can all be highly significant. These cues can be difficult to detect and interpret; web pages are an interesting domain for experimenting with methods for using these cues. Third, many web pages are very complex and have many regions in a deep hierarchy. The latter two factors make the segmentation problem challenging despite the structural assumptions that can be made about regions.

The use of a Bayesian model for the segmentation method is motivated by several factors. It allows the use of a sophisticated prior probability distribution over segmentations based on the expected structure of a web page. Although the experiments shown in this thesis achieve good results using relatively simple prior probability distributions, the possibility of learning a detailed model of web page structure is an important feature of the Bayesian approach and we hope to investigate this further in the future (see Section 8.3.4 on image statistics). The structural assumptions that can be made about the shape of regions make the domain of web pages ideal for experiments with a Bayesian approach, since these assumptions simplify the problem relative to the much more complex region shapes found in natural images. Finally, a Bayesian approach allows easy integration of different types of evidence. This is important because a practical assistive interface has the potential to realize better performance by combining evidence from the DOM tree with evidence from the image of the rendered page. With an arbitrary energy function, adding DOM tree evidence would require extensive modification of the algorithm; in an explicitly probabilistic Bayesian framework, multiple sources of evidence can be expressed in a common probability measure and combined according to well-established principles. A Bayesian approach requires the estimation of prior probabilities at each stage of the segmentation process; we discuss these prior probability distributions as each becomes relevant.

3.2 Early Attempts

In this section we describe the first of two segmentation algorithms developed in the course of my thesis research.

3.2.1 Ontology

In the first model we developed, a web page is represented by a tree of regions. The root of the tree is a region consisting of the entire page, leaf nodes are "simple" regions

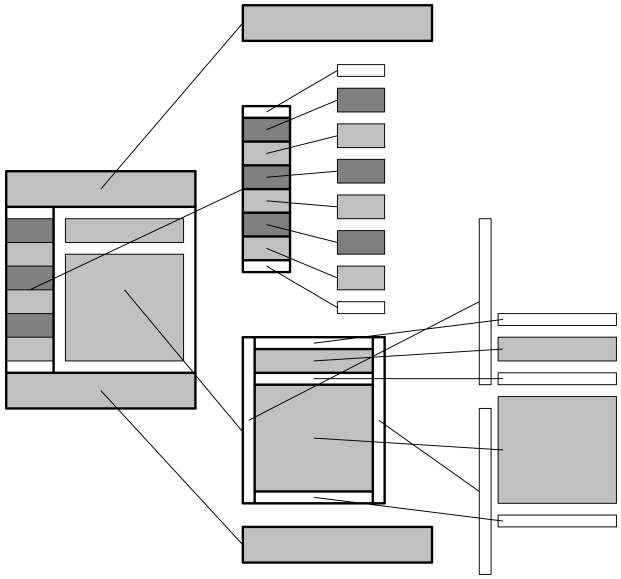


Figure 3.1: Simplified example of a segmentation tree, showing the division into distinct regions at different levels.

which needn't be divided further, and internal nodes represent "complex" regions to be subdivided or used as a whole, depending on the requirements of the application. The locations of the divisions between sibling regions in the tree are determined by evidence provided by the locations of edges in the image of the parent region. Figure 3.1 shows an example of this type of segmentation tree for a simplified, abstract web page.

An edge in an image of a web page may be caused by a line across a background (a ridge or trough edge), by a difference in two colours (a step edge), by an alignment of objects (an alignment edge), or by the boundaries of the page image (a boundary edge)[2]. The first three of these types are shown in Figure 3.2. Alignment edges are composed of aligned ridge or step edge segments corresponding to the edges of the objects in alignment[3]. Ridge or trough edges may be caused by the use of a line to divide sections with the same background colour, as in the case of the <hr> (horizontal rule) HTML tag, or the lines dividing cells in a table. Step edges may be caused by the use of different background

[2]Natural images may also contain roof edges, which are steps in the first derivative of the image intensity. We do not consider these explicitly here, since they do not play a significant role in common web design conventions as step and ridge edges do.

[3]It is also worth noting that alignment edges are similar to illusory contours [127], although an alignment edge need not necessarily produce a perceived contour in the gaps between edge segments.

(a) Ridge edge (`www.td.com`)

(b) Step edge (`www.facebook.com`)

(c) Alignment edge (`www.reddit.com`)

Figure 3.2: Types of edges commonly found in web pages, illustrated with examples from real web pages.

colours to set regions apart. An alignment edge may be caused by the organization of objects into regions, as in the case of aligned lines of text. Each of these types may, of course, have a less-significant cause. Finally, the edges or borders of the image of the page must be considered significant—outside of the page is clearly semantically different from inside the page. This is not the case for natural images, where the borders of the image are not determined by the scene but by the limitations of the camera. In some specific cases, such as artistic photographs, the photographer may deliberately align the image boundaries with semantically significant features of the scene, but this is not the same as the case of web pages, in which the image borders *define* the boundaries of the "scene".

Edges in the page image provide evidence for the boundaries of regions. The mere presence of an edge does not, however, guarantee the presence of a true boundary. Region boundaries must also satisfy an additional set of criteria for validity:

1. A region boundary must be closed
2. A region boundary must be rectangular
3. A region boundary must be axis-aligned

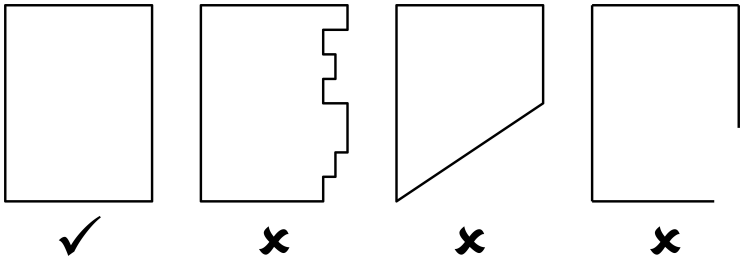


Figure 3.3: Valid and invalid regions under our definitions. The first region is valid, as it is defined by a closed, axis-aligned, rectangular contour. The second is not rectangular; the third is not rectangular and one edge is not axis-aligned; the fourth is not closed.

While these criteria would be far too limiting for image segmentation in general, they are reasonable in the context of web pages. The elements of a web page are defined on an axis-aligned grid, and it is unusual to encounter web page elements for which a rectangular bounding box would be a poor representation. Regions described by axis-parallel bounding boxes are convenient to implement, as the CSS box model [66], a key part of the ontology of CSS and therefore critical to the process of designing web pages, operates on axis-parallel rectangular boxes. The DOM tree itself, used in the page rendering process, represents page elements as a tree of regions with axis-parallel rectangular bounding boxes. Figure 3.3 depicts regions that follow and violate these rules.

The page is modelled as a hierarchical tiling of regions. This tiling must obey the following four rules:

1. Each region in the tiling must be a valid region as described above
2. Each region must have at most one parent region
3. Every region but the root (which occupies the entire page image) has exactly one parent region
4. Every pixel in a parent region must be a member of exactly one child region

Examples of valid and invalid tilings are shown in Figure 3.4. These rules enforce a tree structure on the segmentation of the page; thus, it is a hierarchical segmentation of the page image.

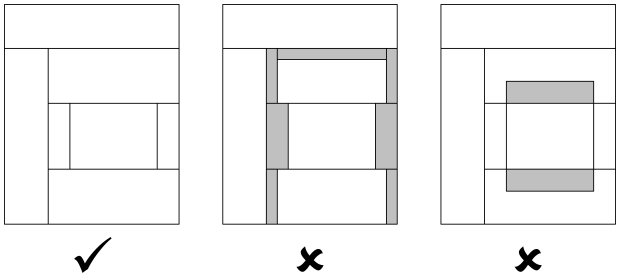


Figure 3.4: Valid and invalid tilings under our definitions. The first is valid, as it covers the entire page without overlapping regions. The second is invalid because it does not cover the entire region (uncovered regions are shown in grey); the third, because regions overlap each other (overlapping regions are shown in grey).

3.2.2 Edge Detection

The first step in segmenting the page is edge detection. The segmentation algorithm uses a Bayesian method to determine the probability of a locally significant edge in a specified direction passing through a each pixel. The edge detection method finds both ridge and step edges directly; alignment edges are not detected directly, but alignments are considered as evidence in the construction of region boundaries. Our method is based on finding possible edges that "stand out" from their surroundings in the map of Sobel edge strengths[4]; this is intended to account for the presence in web pages of both highly textured regions such as text blocks where strong edges are common but do not divide semantically significant regions, and subtle but semantically significant edges such as a small change in background colour. In other words, it is necessary to have a strong mechanism for adapting the estimate of the probability that an edge is locally significant to local conditions in the page. Figure 3.5 shows the edge detection process, as described in this section[5].

To determine if a potential edge pixel stands out from its surroundings, we must first define what those surroundings are. The simplest neighbourhood (*i.e.* immediate surroundings) would include all of the pixels within a specified distance of the point in question. This would, however, include points along the proposed line; if the line exists, these points would paradoxically make the evidence for the existence of the line seem weaker by adding more strong edges in the neighbourhood. Excluding these points prevents this problem.

[4]Sobel edge filters are local first derivative operators commonly used as a first step in edge detection [25]

[5]Note that the edge strength distributions are shown as probability density functions, rather than cumulative distribution functions, since this makes the presence or absence of small features clearer.

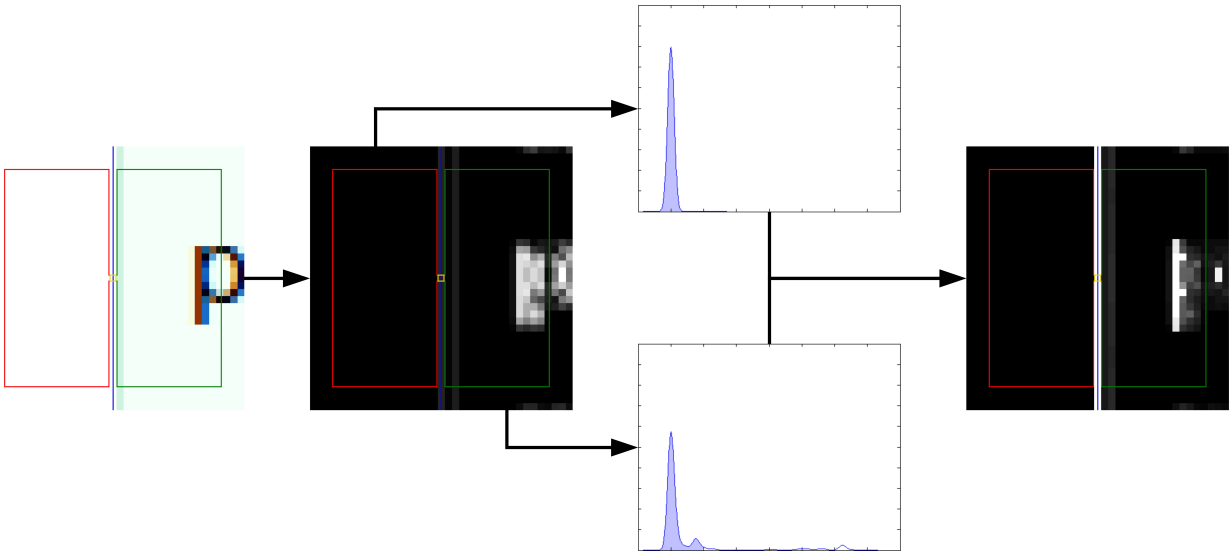


Figure 3.5: An illustration of the process of calculating the probability of a vertical edge at the central pixel. The two neighbourhoods on either side of the proposed line are outlined in red and green, with the central pixel whose edge probability is being calculated outlined in yellow. The first (leftmost) image shows a patch from the original page image. The second shows preliminary Sobel edge strengths for the patch. The two stacked images in the third column show the distributions of preliminary edge strengths in each neighbourhood. The final image shows a map of edge probabilities; the central pixel is the edge probability calculated, which is here shown with the probabilities of nearby pixels as well. The edge strength distributions are obtained using kernel density estimation. Shaded regions correspond to edge strengths less than that at the yellow-outlined pixel.

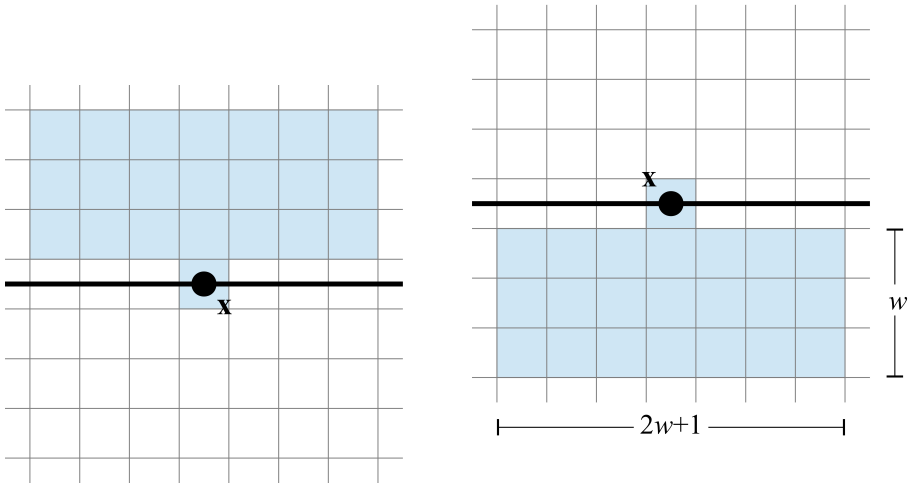


Figure 3.6: Neighbourhood structure as currently implemented. The thick line shows the location of a possible horizontal line and the large dot the point whose edge strength is being evaluated. The shaded regions show the neighbourhoods on each side of the line. The half-window width parameter w controls the size of the neighbourhood. For vertical lines, this structure is rotated 90°.

For an edge between two textures, where one contains many short edges and the other contains very few, the edge "stands out" more relative to the texture with few edges than relative to the texture with many. It is therefore worth examining pixels on each side of the proposed line *separately* (*i.e.* as two distinct neighbourhoods), and considering the degree to which the edge "stands out" from either side to be the true degree to which it stands out from its surroundings. To limit the degree to which the pixel in question can stand out from the neighbourhood, we include each pixel in its own neighbourhood; this has been found to give good results in practice and avoids numerical issues later. Figure 3.6 shows our neighbourhood structure.

Having defined the appropriate neighbourhoods around a possible edge, we must define what it means for an edge to stand out from it. For a page image I with three colour channels, (I_R, I_G, and I_B), we define a map of the preliminary edge strength for each direction. The map for horizontal edge strength is defined, in the continuous case, by:

$$S_h(x,y) = \sqrt{\frac{\partial I_R}{\partial y}(x,y)^2 + \frac{\partial I_G}{\partial y}(x,y)^2 + \frac{\partial I_B}{\partial y}(x,y)^2} \tag{3.1}$$

The vertical preliminary edge strength S_v is defined analogously[6]. When the same equation applies to both the horizontal case and the vertical case, we use $S(x, y)$ to indicate that either edge strength may be used. In the discrete case, we use the vertical and horizontal Sobel partial derivative kernels [25]. Because this is an initial processing step rather than a single-shot detection of edges that will be used directly, a simple convolution-based edge detection method, rather than a more sophisticated method such as [23] or [106], was selected.

To determine if a possible edge "stands out" we must have a description of the surroundings to compare it to. Since the objective is to determine if a potential edge pixel is in fact on a locally significant edge, we compare its strength to the edge strengths at all neighbouring pixels. We describe these strengths using a probability distribution $P(e)$ over edge strengths e in a neighbourhood $N(x, y)$.

The null hypothesis is that the edge at a given pixel does not stand out from its surroundings. It is important to note that this is not equivalent to a null hypothesis that there is no edge at all; an edge can exist, but still not be locally significant because it is surrounded by other strong edges. It is therefore not the absolute strength of the edge that determines its significance, but its strength relative to edge pixels in its neighbourhood. We assume that if the null hypothesis holds, the edge strengths at the pixels in the neighbourhood (including the pixel being examined) are independent and identically distributed according to the distribution P. Figure 3.7 shows the distribution of gradients in pairs of adjacent pixels. In Section 8.3.2, we revisit the possibility of using more sophisticated models which relax these independence assumptions. It is also worth noting that the neighbourhood size should be selected such that edges should be separated by a distance comparable to the window size in order to be considered locally significant. This reduces the probability that the significance of an edge will be underestimated because of another significant edge inside the window.

The simplest approach would be to assume a normal distribution for $P(e)$ and determine its parameters from the neighbourhood edge strengths, but the assumption of normality produces poor results in practice due to the highly non-normal distributions of edge strengths. Instead, we use a nonparametric estimate of the distribution of edge strengths in the neighbourhood. Applying kernel density estimation (with a Gaussian kernel) to the observed distribution of edge strengths in the neighbourhood, we obtain a

[6]The image is not blurred before estimating the local derivative because the web page rendering process does not produce high-frequency noise in the image of the page, and the scale of the page as rendered is its natural scale.

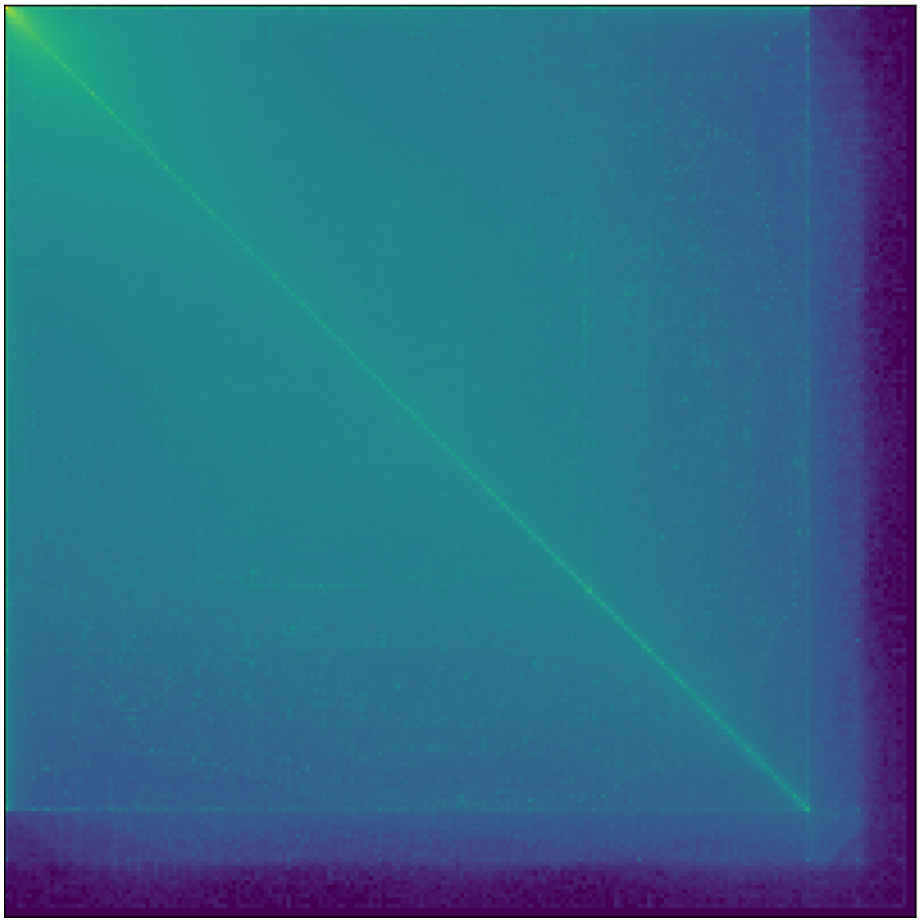


Figure 3.7: Histogram of adjacent pairs of gradients in a dataset of 100 web pages. The vertical and horizontal axes represent gradient values in the pair; colour represents the number of observations, on a log scale. Note that the diagonal, representing instances of equal adjacent gradients, is prominent relative to the surrounding area. By far the most common case is equal gradients of zero, represented by the upper-right corner.

probability density function

$$P(s) = \frac{1}{|N|} \sum_{(x,y)\in N} \text{Norm}(s; S(x,y), \sigma) \tag{3.2}$$

where $\text{Norm}(x; \mu, \sigma)$ represents the probability density at x for a normal distribution with a mean of μ and a standard deviation of σ. By using kernel density estimation, we obtain a smoothed, continuous version of the histogram of Sobel edge detector responses in the neighbourhood, suitable for use as a probability distribution. We assume that the Sobel edge detector response at each pixel is independent, and drawn from the distribution P, so that Pr(Sobel response at $(x, y) = s) = P(s)$ for each pixel (x, y) in the neighbourhood. With this probability distribution and these independence assumptions, we can define a probability of observing a pixel with an edge strength of at least $s = S(x, y)$ when no locally significant edge is present given the neighbourhood in which it occurs. For brevity, let $S_{x,y,s}$ represent the event $S(x, y) \geq s$, $E_{x,y}$ the event "the pixel at (x, y) is a locally significant edge pixel", and P represent the neighbourhood edge strength distribution defined in Equation 3.2. Then

$$\Pr\left(S_{x,y,s} \left| \overline{E_{x,y}}, P\right.\right) = 1 - CDF_P(s) = 1 - \int_0^s P(t)\mathrm{d}t \tag{3.3}$$

where CDF_P represents the cumulative distribution function of the distribution P. By Bayes' Theorem,

$$\begin{aligned}
\Pr\left(\overline{E_{x,y}} \left| S_{x,y,s}, P\right.\right) &= \frac{\Pr\left(S_{x,y,s} \left| \overline{E_{x,y}}, P\right.\right) \Pr\left(\overline{E_{x,y}} \middle| P\right)}{\Pr\left(S_{x,y,s} | P\right)} && (3.4)\\
&= \frac{\left(1 - \int_0^s P(t)\mathrm{d}t\right) \Pr\left(\overline{E_{x,y}} \middle| P\right)}{\Pr\left(S_{x,y,s} | P\right)} && (3.5)\\
&= \frac{\Pr\left(\overline{E_{x,y}} \middle| P\right)}{\Pr\left(S_{x,y,s} | P\right)} - \frac{\left(\int_0^s P(t)\mathrm{d}t\right) \Pr\left(\overline{E_{x,y}} \middle| P\right)}{\Pr\left(S_{x,y,s} | P\right)} && (3.6)
\end{aligned}$$

Every region has edges. Considering an arbitrary point in a region, without considering its own preliminary edge strength, the distribution of edge strengths in its neighbourhood does not inherently affect the probability that the point in question is an edge. We therefore assume that $\Pr\left(\overline{E_{x,y}} \middle| P\right) = \Pr\left(\overline{E_{x,y}}\right)$ (*i.e.*, the probability that a given pixel in a region is an edge pixel does not depend on the distribution of preliminary edge strengths in the region, in the absence of evidence about the relationship between this distribution and the

preliminary edge strength of the pixel in question). Thus

$$\Pr\left(\overline{E_{x,y}}\,|S_{x,y,s},P\right) = \frac{\Pr\left(\overline{E_{x,y}}\right)}{\Pr\left(S_{x,y,s}|P\right)} - \frac{\left(\int_0^s P(t)\mathrm{d}t\right)\Pr\left(\overline{E_{x,y}}\right)}{\Pr\left(S_{x,y,s}|P\right)} \tag{3.7}$$

$$= \frac{\Pr\left(\overline{E_{x,y}}\right)}{\Pr\left(S_{x,y,s}|P\right)}\left(1-\int_0^s P(t)\mathrm{d}t\right) \tag{3.8}$$

Further algebraic manipulation yields

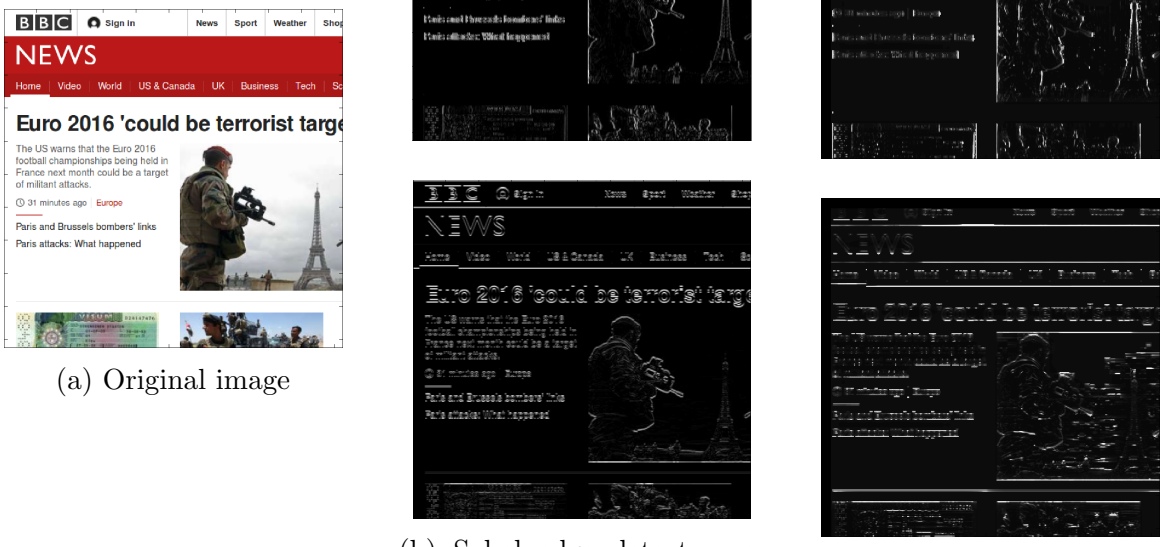
$$\Pr(E_{x,y}|S_{x,y,s},P) = \frac{\Pr\left(E_{x,y}\right)\left(1+\Pr\left(S_{x,y,s}\left|\overline{E_{x,y}},P\right.\right)-\Pr\left(E_{x,y}\right)\Pr\left(S_{x,y,s}\left|\overline{E_{x,y}},P\right.\right)\right)}{\Pr\left(E_{x,y}\right)+\Pr\left(S_{x,y,s}\left|\overline{E_{x,y}},P\right.\right)-\Pr\left(E_{x,y}\right)\Pr\left(S_{x,y}\left|\overline{E_{x,y}},P\right.\right)} \tag{3.9}$$

This represents the probability of an edge at the image position (x, y), given the preliminary edge strength and the statistical distribution of edge strengths in the immediate context of (x, y). In this equation, the prior probability of an edge $P(E_{x,y})$ is an important parameter, which represents the expected density of edges. A high value of $P(E_{x,y})$ encourages the belief that a given pixel has a locally significant edge.

Figure 3.8 shows the results of our edge detection algorithm compared to the original Sobel edge detector responses. Note that while the text is full of strong edges in the Sobel edge detector response, in the maps of the probability that each pixel contains a locally significant edge, the edges of the text block are emphasized and the middle of the text block is suppressed. This is even true for the large text of the headline. Note also that the faint edge at the top of the photograph shows negligible edge strength in the original edge detector responses, but is clearly visible in the edge probability map. This example clearly shows that our method for estimating locally significant edge probablities from edge detector responses is able to account for context.

3.2.3 Probability of a Line

A segmentation of a page is supported, in our model, by semantically significant lines along the divisions between regions. In order to determine the probability that a proposed segmentation is correct, it is therefore necessary to first define the probability that a proposed line between two points in the image is semantically significant. In our model, a line is considered to be semantically significant if and only if each pixel in that line is a locally significant edge pixel. Assuming that the probabilities of local significance are mutually independent for all pixels on the line, the probability that a semantically



(a) Original image

(b) Sobel edge detector responses

(c) Edge probability maps

Figure 3.8: Comparison of original image, Sobel edge detector responses, and a map of the probability of a locally significant edge.

significant horizontal line exists between (x, y) and (x', y) given the image data I is

$$\Pr(\mathrm{L}_{(\mathrm{x,y}),(\mathrm{x}',\mathrm{y})}|\mathrm{I}) = \prod_{t=x}^{x'} \left(1 - \Pr\left(\overline{E_{t,y}} \middle| S_{t,y,s}, N(t,y)\right)\right) \tag{3.10}$$

where $N(t, y)$ represents the neighbourhood in I around the pixel (t, y). Vertical line probabilities are defined analogously.

3.2.4 Probability of a Segmentation

We evaluate the quality of a proposed segmentation of a web page by estimating the probability that the segmentation is supported by the visual appearance of the page. There are two significant factors in the quality of a segmentation:

- There should be evidence supporting all divisions that are made
- There should be no evidence of additional divisions across the entire width or height of the region

The overall probability that the evidence supports a given level of a segmentation is the probability that semantically significant lines exist along all divisions present, and that no such lines cross entire regions (as a line crossing a region in the absence of a division in the segmentation would indicate that a division should have been placed along the line). This does not fully account for the support given to the presence of a line by the presence of other high-probability segments along the line, but it was found to be a useful heuristic in practice. All lines are assumed to be independent. The algorithm for calculating segmentation probabilities is shown in Algorithm SegmentationQuality.

3.2.5 Finding a Segmentation

Our first segmentation algorithm is shown in detail in Appendix A—we provide an overview here. In order to find the most probable segmentation tree for an entire page, we recursively divide each region. For each region that is to be divided into child regions, we optimize locally over the segmentation quality q found by Algorithm SegmentationQuality in order to find the most probable segmentation of that region. Starting from a region consisting of the entire image, we optimize recursively until either a predefined maximum tree depth

Input: Vertical and horizontal edge probabilities E_v and E_h (obtained from Equation 3.7), set S of regions
Output: Probability-based quality measurement $q \in [0, 1]$
Probability of a false negative division $p_{fn} \leftarrow 1$;
Probability of a false positive division $p_{fp} \leftarrow 1$;
foreach $R \in S$ **do**
  Let HE and VE represent the sets of the coordinates of pixels in the horizontal and vertical edges of R, respectively;
  $p_{fp} \leftarrow p_{fp} \times \left(1 - \prod_{(x,y)\in HE} E_h(x,y) \times \prod_{(x,y)\in VE} E_v(x,y)\right)$;
  $p_{fn} \leftarrow p_{fn} \times \prod_{x=R_{xmin}}^{R_{xmax}} \left(1 - \prod_{y=R_{ymin}}^{R_{ymax}} (1 - E_v(x,y))\right)$;
  $p_{fn} \leftarrow p_{fn} \times \prod_{y=R_{ymin}}^{R_{ymax}} \left(1 - \prod_{y=R_{xmin}}^{R_{xmax}} (1 - E_h(x,y))\right)$;
end
return $q = (1 - p_{fn})(1 - p_{fp})$

Function SegmentationQuality(E_h, E_v): Algorithm for calculating the quality of a proposed segmentation. R_{xmin} and R_{xmax} represent the minimum and maximum horizontal positions of internal pixels in a region r; R_{ymin} and R_{ymax} are defined analogously.

is reached or a region has no logical division points. The current implementation of the system uses a simple but effective greedy approximate optimization method. First, a grid of "candidate boundaries" is found, corresponding to rows or columns in the image with particularly strong evidence for a division. Second, the optimization algorithm searches for the segmentation best supported by the evidence from the image, subject to the constraints that region boundaries must lie on candidate boundaries and the number of child regions is not excessively large.

Candidate boundaries are detected by measuring the strength of the evidence for a line crossing the image horizontally or vertically at each position. For a horizontal candidate boundary, $CB(y) = \prod_{t=0}^{\text{image width}} \left(1 - \Pr\left(\overline{E_{t,y}} \middle| S_{t,y,s}, N(t,y)\right)\right)$, as described in Section 3.2.4. To account for near-alignments of edges, we convolve the sequence $CB(y)$ with a discrete Gaussian kernel with a standard deviation of 1 pixel, truncated at $\pm 3\sigma$. We use outlier detection to find candidate boundaries. If $CB(y)$ is sufficiently large, it is an outlier, but the threshold for "sufficiently large" must depend on the distribution of candidate boundary strengths in CB. To account for variation in the average candidate boundary strength, we subtract the mean of CB, and to account for variation in the variance of CB we divide this difference by the standard deviation to find the distance of $CB(y)$ from the mean in units equal to the standard deviation of CB; that is, if $\frac{CB(y) - \text{mean}(CB)}{\text{stddev}(CB)} \geq t_{CB}$

for some threshold of significance t_{CB}, we assume that there is a candidate boundary at position y. We use an initial value of $t_{CB} = 3$, set as a reasonable value that performed well on an older dataset of 30 images. Although we use the evidence for a line across the full width or height of the image to find candidate boundaries, it is reasonable to expect that lines that do not fully cross the region but have strong evidence for a significant length will appear in this set as well.

The sets of vertical and horizontal candidate boundaries taken together form a candidate boundary grid, which is much coarser than the resolution of the image, dramatically reducing the search space. If the candidate boundary grid is still too large to search (*i.e.*, too many lines qualify as candidate boundaries), the threshold t_{CB} used to detect peaks in the strength of evidence for an edge is increased by δ_t (in our implementation a moderate value of 0.5), progressively, until the candidate boundary grid is sufficiently small. Even for candidate boundary grids at a reasonable size, an exhaustive search of all segmentations is not feasible. Since it is also undesirable for a region to have a large number of child regions, only solutions with a reasonable number of child regions are checked; this constraint encourages solutions that correspond to a reasonable intuition about page structure[7] and reduces the search space still further. Note that if the true number of semantic children is larger than this child region threshold, some will simply be grouped together in the next layer; they could then be separated in the next layer if there is sufficient evidence. Since long list-like structures are common in web pages, we increase the permissible number of child regions for candidate boundary grids that have a single row or column of cells. This results in two sets of lists of possible segmentations:

- For $x \times y$ rectangular grids of cells such that $x, y \geq 2$ and $x, y \leq 35$, we find all segmentations with a maximum of 8 child regions
- For $1 \times n$ or $n \times 1$ grids for $n \leq 23$, we find all segmentations with a maximum of n child regions

Since the sets of tilings depend only on the size of the candidate boundary grid and not on the contents or proportions of the cells, the sets can be pre-computed. This improves performance, although the current testbed implementation is not, in general, highly optimized.

[7] By avoiding a segmentation with a flattened hierarchy with little structure, which may be more difficult for users to navigate.

3.2.6 Evaluation Dataset

This segmentation algorithm was evaluated both qualitatively and quantitatively. Qualitative results are shown first, followed by a quantitative comparison between segmentations produced by our algorithm and segmentations produced by taking the bounding boxes of the nodes of the DOM tree of the page. Before discussing the evaluation, however, it is useful to discuss the dataset itself.

The test dataset used in the experiments described here consists of images of the top 50 web sites in Canada (as ranked by Alexa[8]). The images were taken of the main site, possibly after clicking through an introductory page (such as selecting the English version of Wikipedia) but not bypassing any login pages (because active accounts were not available for most web sites that requested the the user log in to continue). The images were collected on October 9, 2015. The full list of web sites is shown below. The dataset is diverse, including news, shopping, social networking, and business pages. Each page was rendered in Firefox 18.0.1 for Ubuntu, and the page image captured using the extension Screengrab (fix version) 0.98.09c. A window width of 1024 pixels is used for all images. They were segmented using our method implemented as an offline testbed system running in MATLAB. The page images are approximately 1000 pixels wide and range from approximately 600 pixels to over 10000 pixels in height.

1. google.ca
2. google.com
3. facebook.com
4. youtube.com
5. yahoo.com
6. amazon.ca
7. wikipedia.org
8. live.com
9. twitter.com
10. amazon.com
11. kijiji.ca
12. linkedin.com
13. reddit.com
14. netflix.com
15. imgur.com
16. msn.com
17. cbc.ca
18. instagram.com
19. royalbank.com
20. pinterest.com
21. ebay.ca
22. apple.com
23. bing.com
24. td.com
25. paypal.com
26. imdb.com
27. ebay.ca
28. tumblr.com
29. craigslist.ca
30. wordpress.com

[8] www.alexa.com

31. diply.com
32. theweathernetwork.com
33. rbcroyalbank.com
34. cnn.com
35. microsoft.com
36. bmo.com
37. cibc.com
38. kat.cr
39. bestbuy.ca
40. theglobeandmail.com
41. scotiabank.com
42. go.com
43. vice.com
44. indeed.com
45. gfycat.com
46. stackoverflow.com
47. aliexpress.com
48. dropbox.com
49. walmart.ca
50. wikia.com

For the following tests, the segmentation algorithm was run with a threshold for candidate edge significance $t_{CB} = 3$ (for its initial value, with an update increment $\delta_t = 0.5$), an *a priori* probability of an edge $\Pr(E_{x,y}) = 0.01$, a neighbourhood size of $w = 15$, and a kernel width for kernel density estimation of edge strength distributions of $\sigma = 0.1$. The maximum tree depth is set to 6 in the following experiments, except where otherwise noted[9].

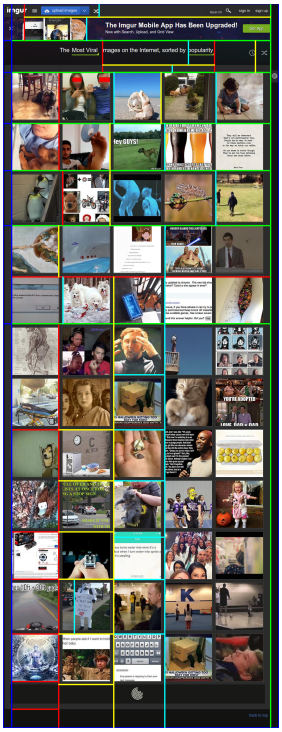
3.2.7 Qualitative Evaluation

The qualitative results we present show selected examples of full and partial segmentations. We show examples illustrating the advantages of our segmentation method and examples showing cases that are difficult for our algorithm to handle.

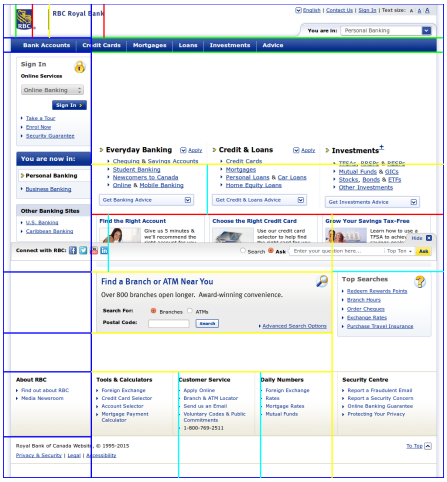
Some example segmentations are shown in Figure 3.9, one from a news web site, one from an entertainment website, and one from a banking website. It is clear that the divisions are largely reasonable, especially for column divisions; even a grid structure is detected fairly well despite the large number of nodes in it. These examples were all generated using entire web pages, not just the area visible on the screen at any given time. As width is typically fixed by the width of the browser window, but length is variable due to vertical scrolling, the full web pages tend to be long and narrow. Figure 3.10 shows details from the two images on the left at a higher resolution.

[9]The neighbourhood size w should be set to a moderate value, larger than the gap between two lines of text in the same paragraph, but not so large as to cross major divisions; we chose 15 pixels for this value. The maximum tree depth is set to a reasonable value to avoid excessively detailed segmentations. Good values for other parameters were determined experimentally on an earlier dataset of 30 diverse web pages, to avoid overfitting.

(a) `yahoo.com`

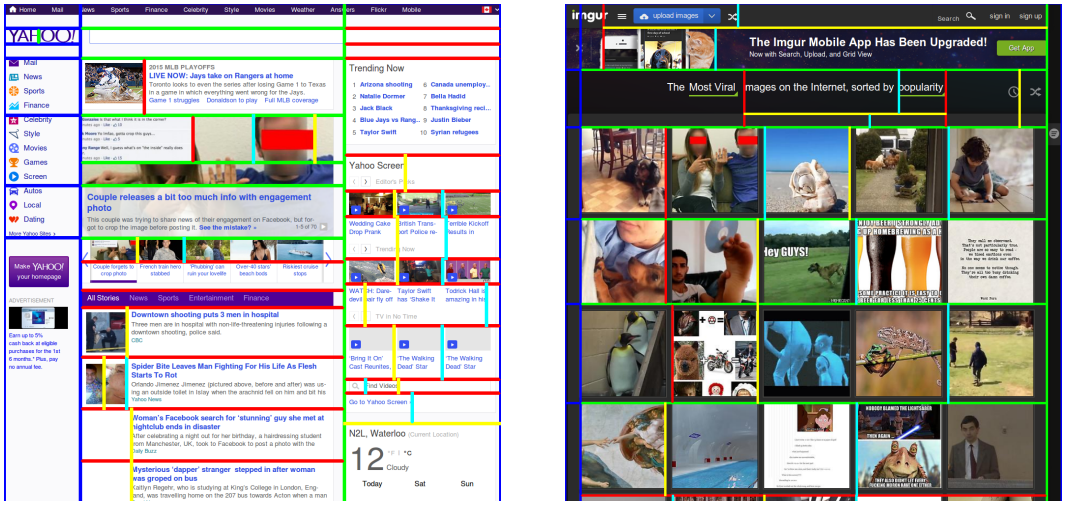


(b) `imgur.com`



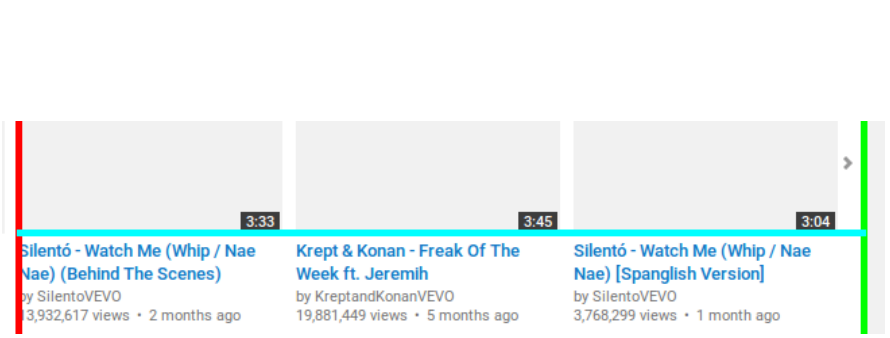
(c) `rbcroyalbank.com`

Figure 3.9: Our proposed method applied to three different web pages, showing the quality of the resulting segmentation. The divisions are colour-coded to represent the levels in the hierarchy; in order from top-level down, the colours are blue, green, red, yellow, and cyan.



(a) `yahoo.com` (b) `imgur.com`

Figure 3.10: Detail images from Figures 3.9a and 3.9b, showing segmentations at a higher resolution.



(a) Faint edge

(b) Sparse edge

Figure 3.11: Successful detections of two difficult edges: one faint, at left (from `youtube.com`), and one sparse, at right (from `cbc.ca`).

(a) A coincidental alignment of edges within text (the letter "K") and the edge of a thumbnail, detected as a spurious low-level edge. The underlying web page is the IMDB home page.

(b) Imperfect detection of an overlay in the center of a background image, showing the difficulty with this case for our model, which does not include overlays. Detail from Bing home page.

Figure 3.12: Examples of failure cases for our segmentation algorithm.

Figures 3.11a and 3.11b show successful detections of two difficult edges. In Figure 3.11a, the edge is a faint transition in background colour between light grey and white, but it is detected correctly because it stands out from its immediate surroundings, which contain no other edges. In Figure 3.11b, the edge is formed by the vertical alignment of several regions of text. Although the text produces strong edges on one side of the edge, the other side has no texture nearby, resulting in a correct detection despite gaps between the blocks.

Figure 3.12a shows an example of a spurious division caused by a coincidental alignment of the vertical line in the letter "K" and the vertical edge of a thumbnail. It is useful to segment the thumbnail separately, but the coincidental alignment causes the division to extend too far.

Figure 3.12b shows a flawed detection of an overlay on an image. Perfect detection of overlays is not realistic, since they constitute a violation of the assumption that the regions in the page are rectangular and non-overlapping. Extensions to our model may allow it to handle such overlays correctly by admitting segmentations in which the child node or nodes of a region do not fully cover their parent region.

(a) Original image (b) DOM tree bounding boxes

Figure 3.13: Example of an extraneous edge in the DOM tree (the edge on the far left) which is not supported by visual divisions in the original image. Images from `diply.com`.

While spurious edges can occur in our segmentations, a segmentation with boundaries corresponding to the bounding boxes in the DOM tree can also produce edges without visual support. An example is shown in Figure 3.13. The leftmost edge is not supported by image data; it is produced by a functional region whose edges are not visually distinct from its immediate surroundings.

3.2.8 Quantitative Evaluation

The purpose of the experiments described in this section is to compare our segmentation tree to competitor algorithms. It is especially desirable to compare to a method based purely on the DOM tree, to contrast with the image-based approach. One obvious choice for such a competitor is the DOM tree itself. An initial study was performed using the segmentations produced by all bounding boxes of elements in the DOM tree. Since the DOM tree is generally not used in its "raw" state for page segmentation in practice, we also compared our segmentation method quantitatively to two existing algorithms: a "control" version of the visual segmentation algorithm which uses evidence from the DOM tree rather than from the image of the page, and the well-known VIPS algorithm [22] (described in Sections 2.2.1 and 7.3). The comparison with the control algorithm is useful for comparing segmentations produced using evidence from the image to those produced using evidence from the DOM tree, using the same optimization methods and tiling-based segmentation model. The comparison with VIPS shows the relationship between our method and one of

the most familiar web page segmentation algorithms.

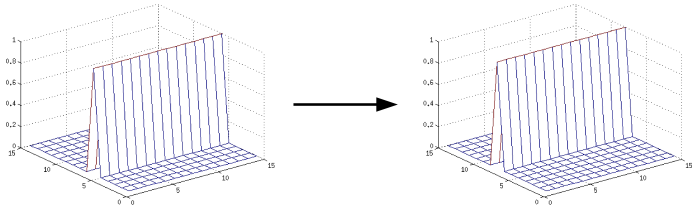
If our algorithm produces better results, according to some reasonable metric, or even comparable results (given the greater generality of our method), then we can demonstrate that our proposed approach is valuable and thus offer an initial validation. In the experiments described here, we use several methods of comparison, including evaluating segmentation complexity and agreement with different segmentation algorithms. For determining agreement, the "flat" segmentations produced by the leaf nodes are used. These measures are intended to be used in combination with the qualitative results shown in Section 3.2.7.

The control algorithm is a variation on our segmentation algorithm which uses information from the DOM tree (collected using a purpose-built Firefox extension) to perform the segmentation, rather than information from the image. Specifically, we replace the edge map with a "bounding box map", where the probability that pixel x is a horizontal edge is defined to be p_{bb} if it is on a horizontal edge of a bounding box, and p_{nonbb} otherwise (p_{bb} and p_{nonbb} are parameters set to reflect uncertainty about the relationship between the presence of a bounding box edge and the presence of a semantic boundary); vertical edge probabilities are defined analogously. For our experiments, we use parameter values $p_{bb} = 0.9$ and $p_{nonbb} = 0.01$ to reflect the intuition that semantic boundaries are likely but not certain to occur on or near DOM tree node bounding boxes, and possible but unlikely to occur elsewhere. The use of this algorithm provides better experimental control than using an existing competitor would; since the optimization method and segmentation model are identical, any difference in the results must derive from the use of the DOM tree bounding boxes in place of image data.

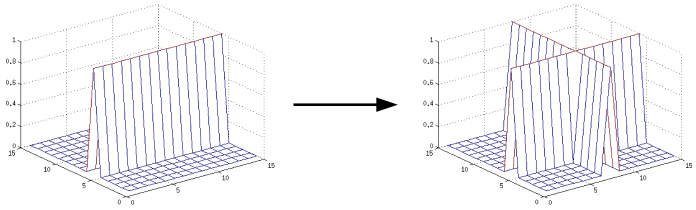
For our comparison with VIPS, we use the reference implementation found at `http://www.cad.zju.edu.cn/home/dengcai/VIPS/VIPS.html`, and specifically the "Using VIPS" application (slightly modified to ensure that the rendering window is the same size as used in Firefox to collect screen images). For compatibility reasons, we ran this algorithm in a Windows XP Service Pack 3 virtual machine. The reference implementation uses Internet Explorer to render the page and obtain the style and placement information used by the VIPS algorithm; we use Internet Explorer 8, as it is the most recent version compatible with Windows XP. Using this version, three web pages are not correctly rendered; these are omitted from the comparison.

Earth Mover's Distance

To generate data suitable for comparison across segmentation techniques, we create edge maps including all edges across all levels of the segmentation trees. This flattens the



(a) Two edge maps with a small EMD of 30



(b) Two edge maps with a large EMD of 280

Figure 3.14: Examples of the earth mover's distance (EMD) between small sections of synthetic edge maps (where the horizontal plane in the graph represents a section of an image plane, with the vertical axis representing an edge strength). In each case, the edge map on the left is transformed into the edge map on the right by moving mass from one part of the edge map to another (at a cost equal to the product of the amount of mass moved and the distance it is moved) or by adding mass (at a fixed cost per unit of mass set here to 20).

hierarchy, allowing us to avoid the need to compare the structure of trees with differing numbers of nodes and other structural properties, while still preserving all edges between regions.

To compare our segmentations with the segmentations based other methods, we use the earth mover's distance (EMD) between the two edge maps [101, 102]. The EMD describes the total "work" or cost needed to transform one function to another, where work is defined to be the amount of "mass" moved multiplied by the distance moved. Figure 3.14 shows examples of functions that are "close" and "far" according to the EMD. Formally, given two discrete functions $f(\mathbf{x})$ and $f'(\mathbf{x})$, the EMD between f and f' is defined to be

$$d = \min_{m(\mathbf{x},\mathbf{x}'),a(\mathbf{x})} \left(\sum_{\mathbf{x},\mathbf{x}'} c_{\text{move}}(\mathbf{x},\mathbf{x}')m(\mathbf{x},\mathbf{x}') + \sum_{\mathbf{x}} c_{\text{new}}|a(\mathbf{x})| \right) \tag{3.11}$$

where c_{move} represents the cost of moving one unit of mass from $\mathbf{x}$ to $\mathbf{x}'$, c_{new} represents the cost of adding or removing one unit of mass at any given position, $m(\mathbf{x},\mathbf{x}')$ the quantity

	Downsampling	Size	Number of samples
Large-scale	Factor-of-X	Image-dependent	50
Small-scale	None	127×127	500

Table 3.1: Tabular summary of the parameters of the large-scale and small-scale comparisons of segmentation structures using the EMD.

of mass moved from $\mathbf{x}$ to $\mathbf{x}'$, and $a(\mathbf{x})$ the amount of mass added or removed at position $\mathbf{x}$. This is subject to the constraint that

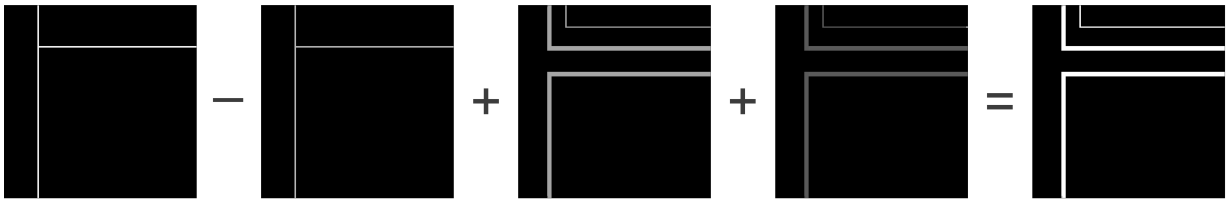
$$f'(\mathbf{x}) = f(\mathbf{x}) + a(\mathbf{x}) + \sum_{\mathbf{x}'} m(\mathbf{x}', \mathbf{x}) - \sum_{\mathbf{x}'} m(\mathbf{x}, \mathbf{x}'). \tag{3.12}$$

In this equation, f and f' again represent the original and target functions. We define $m(\mathbf{x}, \mathbf{x}')$ to be the Euclidean distance between $\mathbf{x}$ and $\mathbf{x}'$. We use a cost of adding or removing mass c_{new} equal to the maximum Euclidean distance between two points in the image, although a lesser cost is possible (we use a cost of 63, half of the patch width, for a richer demonstration of the EMD in Figure 3.15, since this ensures that mass is both added and moved).

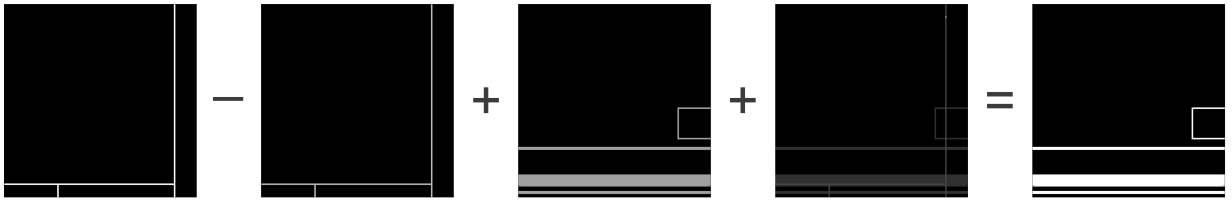
Because the EMD is expensive to compute and the full edge maps are very large, we perform two sets of tests. In the first, randomly-chosen 127×127-pixel patches[10] of the edge maps are compared to each other at full resolution; this is useful for comparison of the small-scale structure of segmentations. We use 10 samples from each page for our experiments. Figure 3.15 shows examples of two such pairs of patches, with different EMD distances, including the components used to obtain the EMD. In the second comparison, the full page edge maps are downsampled (by a factor of 50, so that the areas of large pages are comparable to the area of the patches) while still maintaining the overall structure of the edge maps, and compared to each other directly; this is used for a large-scale comparison of segmentations. Because our visual segmentation algorithm and the DOM tree bounding boxes may produce different numbers of edge pixels, we normalize the edge maps to have the same total mass for each comparison. Table 3.1 compares the two uses of the EMD.

For our purposes, we normalize the EMD by dividing by the maximum cost possible for the given distance function and new-mass cost; this allows easy comparison when, for example, the image size or aspect ratio varies between two cases. We also normalize the

[10]This size was chosen because it is small enough to be practical for large numbers of tests but large enough to encompass interesting features.

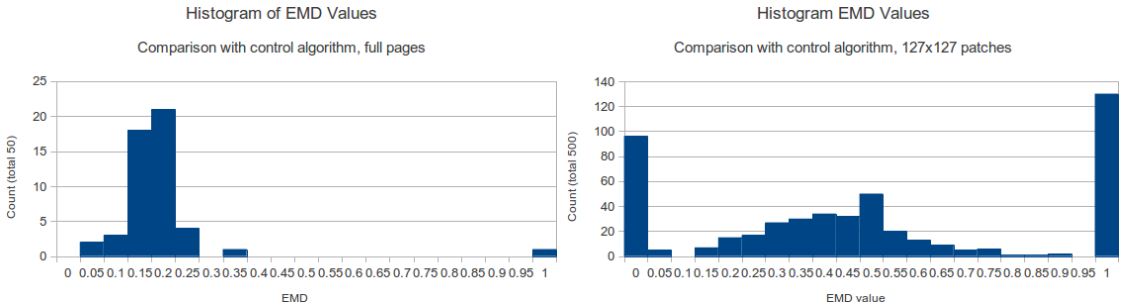


(a) Two patches with an EMD of 90.5



(b) Two patches with an EMD of 143.7

Figure 3.15: Examples of pairs of patches from visual and DOM-based edge maps, showing different levels of similarity. The edge maps are shown as binary images; intermediate stages are shown using the log of the absolute value of mass to better depict structure. The EMD was calculated using an extra mass cost (c_{new}) of 63, half of the patch width.



(a) Full pages (b) 127×127 patches

Figure 3.16: Normalized earth mover's distance between the results of our proposed segmentation method and the results of the control algorithm. The EMD value bins are shown on the horizontal axis; number of occurences of EMD values in each bin are shown on the vertical axis. The left histogram shows the comparison for 50 downsampled webpages; the right shows the comparison for 500 randomly-chosen 127×127 patches. These figures were generated with a new-mass cost c_{new} equal to the largest distance between two points in the image or patch.)

sum of mass in each edge image to 1 (or 0 if the image is uniformly zero) prior to calculating the EMD. The range of the normalized EMD is $[0, 1]$. If the normalized EMD is 0, the two patches or images are identical; if it is small, they are similar; if it is large, they are dissimilar; and if it is 1, they are entirely different (*e.g.* one patch has nonzero mass and the other does not, or all of the mass is concentrated at opposite points in each patch). These extremes often occur for small patches, especially for sparse segmentations.

Comparison with Control Algorithm

The control algorithm provides a useful point of comparison for our algorithm, since it uses the same optimization methods with the same parameters, and only differs from our method in the use of evidence from the DOM tree bounding boxes in place of edges in the image of the rendered page. For this experiment, we use a maximum tree depth (including the root node) of 6. Figure 3.16 shows the differences (expressed in terms of the normalized EMD) between the segmentations produced by our algorithm and the control algorithm.

For downsampled full pages, the mean EMD (with a 95% confidence interval) between the two solutions is 0.167 ± 0.036, assuming a normal distribution, which is reasonable for this data; for patches at full resolution, the mean EMD is 0.478 ± 0.032. This demonstrates

that the two approaches give results that are distinct from each other[11], and more similar at a large scale than at a small scale for $\alpha = 0.005$ (using a t-test, $t(548) = 3.05$, $p = 0.001$). This would be expected if both methods capture broad themes of organization, but treat the details of the structure differently. Furthermore, even at a large scale the two are significantly different. Note also the relatively high occurrence of the extreme values of EMD for small patches, due largely to perfect agreement that an edge does not exist or disagreement about whether or not an edge exists in the patch.

Comparison with VIPS

Our comparison with VIPS uses a slightly reduced dataset of 47 pages. Due to being incompatible with more modern web technologies, VIPS was unable to correctly render three of the webpages; to make the most conservative comparison, we omitted what could not be handled at all in VIPS. We used the default PDoC (permitted degree of coherence) value of 5 to generate the VIPS segmentations; this resulted in a slightly coarser segmentation than the 6-level segmentation trees used above, so we show comparisons with the upper 3 levels and upper 5 levels of our segmentation trees rather than the full depth of the tree. As will be seen, these two tree depths give broadly similar results.

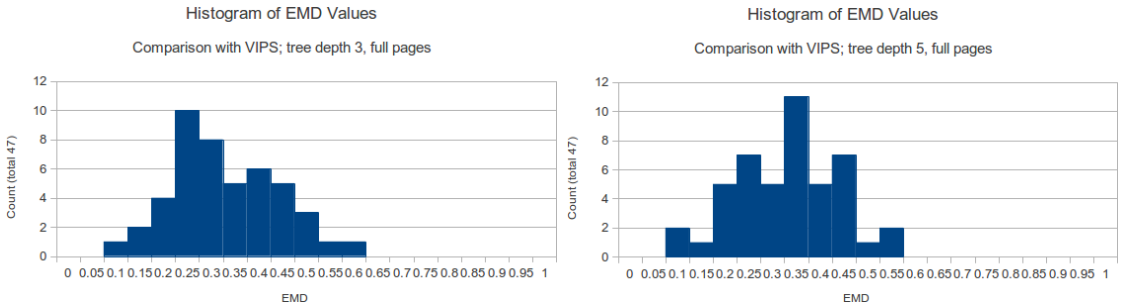
We show results for full webpages in Figure 3.17 and for patches in Figure 3.18. With a tree depth of 3, the mean EMD between downsampled full pages is 0.303 ± 0.032, and the mean EMD between 127×127 patches is 0.449 ± 0.043[12]. With a tree depth of 5, the mean EMD between full pages is 0.308 ± 0.030, and the mean EMD between patches is 0.633 ± 0.040[13]. The smaller tree depth results in better performance in terms of small patches (significant for $\alpha = 0.005$; using a t-test, $t(938) = 3.13$ and $p = 0.0018$), but is not significantly different at a large scale (using a t-test, $t(92) = 0.114$, $p = 0.45$). Both are significantly more different from our segmentations than the control algorithm for full pages[14].

[11] For patches, the mean EMD is significantly different from zero for $\alpha = 0.001$ (using a t-test, $t(499) = 14.6$, $p < 0.0001$). For downsampled complete pages, the mean EMD is significantly different from zero for $\alpha = 0.001$ (using a t-test, $t(49) = 4.64$, $p < 0.0001$)

[12] Both mean EMDs are significantly different from zero for $\alpha = 0.001$ using a t-test. For downsampled full segmentations, $t(46) = 9.47$, $p < 0.0001$; for patches, $t(469) = 10.4$, $p < 0.0001$

[13] Both mean EMD values are significantly different from zero; for downsampled full segmentations, $t(46) = 10.3$ and $p < 0.0001$, and for patches, $t(469) = 15.8$ and $p < 0.0001$.

[14] Using t-tests, at a large scale the mean EMD between VIPS segmentations and our segmentations (with a depth limit of 5) is significantly larger than the mean EMD between control segmentations and our segmentations for $\alpha = 0.001$ ($t(95) = 3.56$, $p = 0.0003$); at a small scale, the difference is significant for $\alpha = 0.005$ ($t(968) = 3.04$, $p = 0.0012$)

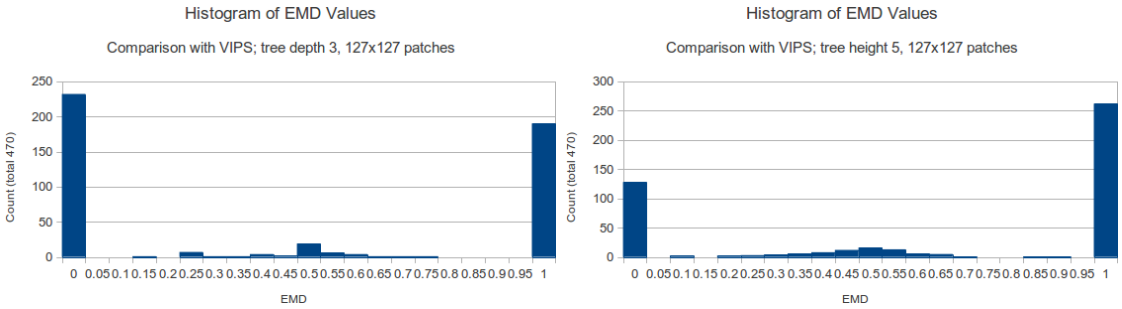


(a) Tree depth 3

(b) Tree depth 5

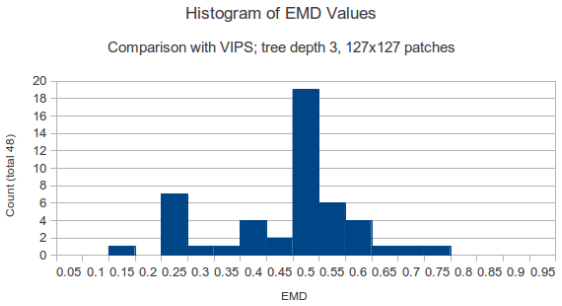
Figure 3.17: Normalized earth mover's distance between the results of our proposed segmentation method (for two different segmentation depths) and the results of VIPS, for 47 downsampled full webpages. Note that the histograms are similar; there is no statistically significant difference in mean distance. These figures were generated with a new-mass cost c_{new} equal to the largest distance between two points in the downsampled image.

Comparing Figure 3.18 with Figure 3.16b, we see that the mean EMD is significantly different from the control algorithm for a tree depth of 5, but not for a tree depth of 3. Qualitatively, an EMD of 0 or 1 occurs much more frequently in the comparison of patches with VIPS than in a comparison of patches with the control algorithm. This is likely to be due at least in part to the relative sparsity of the VIPS segmentation; when no edge appears in either patch, the normalized EMD is zero, and if one patch contains an edge but the other does not, the normalized EMD is one. Because of the prevalence of extreme values in Figures 3.18a and 3.18b, we show histograms of the central region alone in Figures 3.18c and 3.18d, for better viewing of the results. These figures show that while there is partial agreement between the two segmentations, our algorithm still tends to produce results significantly different from VIPS. Demonstrating that our segmentation algorithm produces distinct results does not in itself prove that our algorithm is effective; an algorithm that provides very poor performance could also produce distinct results. The results of our other qualitative and quantitative tests provide support for the quality of our segmentations, and demonstrating that our segmentations are distinct from those produced by VIPS shows that it is not simply a slower means of obtaining the same results.

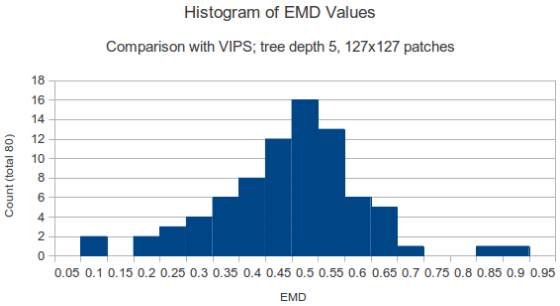


(a) Tree depth 3, full histogram

(b) Tree depth 5, full histogram

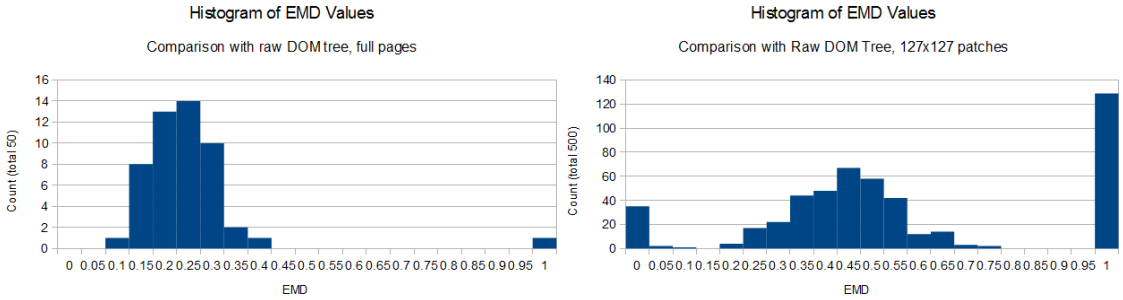


(c) Tree depth 3, center region



(d) Tree depth 5, center region

Figure 3.18: Normalized earth mover's distance between the results of our proposed segmentation method (for two different segmentation depths) and the results of VIPS, for 470 randomly-selected 127×127 patches. Minimum and maximum distance values frequently occur due to agreement that no edge exists in the patch, and occurrence of edges in only one algorithm's segmentation of the patch, respectively. The central regions of the histograms are shown below with rescaled axes. These figures were generated with a new-mass cost c_{new} equal to the largest distance between two points in the patch.



(a) Full pages (b) 127×127 patches

Figure 3.19: Normalized earth mover's distance between the results of our proposed segmentation method and the raw DOM tree. The EMD value bins are shown on the horizontal axis; number of occurences of EMD values in each bin are shown on the vertical axis. The left histogram shows the comparison for 50 downsampled webpages; the right shows the comparison for 500 randomly-chosen 127×127 patches. These figures were generated with a new-mass cost c_{new} equal to the largest distance between two points in the image or patch.

Comparison with Raw DOM Tree

We also performed preliminary tests by comparing our segmentations to segmentations performed by the raw DOM tree (using all bounding boxes of DOM tree nodes as the boundaries of regions) on the same dataset of 50 webpages. As expected, we found that our results are substantially different, especially at a small scale. Figure 3.19 displays these results. For downsampled pages, the distribution of EMD values peaks at approximately 0.2. For patches, the EMD distribution again has peaks at minimal and maximal values. The central peak in this distribution is around 0.5. Especially for patches, these values represent substantial differences.

Simplicity

To evaluate the simplicity of segmentations produced by our method, we compare our method to the segmentation consisting of the bounding boxes of the DOM tree nodes and also to the control algorithm. The former comparison demonstrates that we produce a simpler segmentation than the raw DOM tree does, while the latter demonstrates that our segmentation method does not sacrifice simplicity for versatility. We omit a comparison with VIPS because, with the default PDoC value, it produces a coarser segmentation than

either our method or the control algorithm, and thus the judgment of relative simplicity is less well defined.

We use two measures of simplicity for our comparison: the total number of edge pixels and the total number of basic or bottom-level regions. For our segmentation tree, the bottom-level regions are simply the leaf nodes. The bounding boxes of regions in the DOM tree, however, can overlap (if one box is defined to have a position that partially overlaps another) and are not necessarily covered by their child nodes (whitespace between boxes is possible). Accordingly, we define basic regions in the segmentation based on the DOM tree to be connected regions of non-edge pixels contained by the same set of bounding boxes.

Our segmentation algorithm produces simpler segmentations than the raw DOM tree. In terms of both number of edge pixels and number of bottom-level regions, our method's segmentation is simpler in 49 of 50 cases. The null hypothesis—that the two segmentations are equally likely to be the simpler—can be rejected at a significance threshold of $\alpha < 0.001$ (exact binomial $p = 4.5 \times 10^{-14}$). Thus we can conclude that our segmentation algorithm usually produces a simpler segmentation than the raw DOM tree.

Our segmentation and the control segmentation are approximately equally likely to produce the simpler segmentation as measured by both the number of edge pixels and the number of basic regions; there is no statistically significant difference in probability for either measure. By the number of edge pixels, our segmentation is simpler than the control algorithm's for 24 of 50 web pages; by the number of basic regions, our segmentation is simpler in 25 out of 50 web pages. The null hypothesis cannot be rejected in either case (exact binomial $p = 0.44$ and $p = 0.56$, respectively). This result indicates that our method does not sacrifice the simplicity of the segmentation for independence from implementation details.

Taken together, the results described above indicate that our segmentations are related to the competing segmentations in the following ways:

1. At a small scale, our segmentation produces distinct results from the control algorithm and VIPS (as well as from the raw DOM tree competitor).

2. Our segmentations are more similar to the competing segmentations at a large scale than at a small scale

3. Our segmentation is simpler than the raw DOM tree, in that it presents a coarser display of regions, and is comparable in simplicity to the DOM-based control algorithm

Quantity	Measurement	Interpretation
Similarity between segmentations		
... At a large scale	Normalized EMD between downsampled large images	Small EMD indicates high similarity
... At a small scale	Normalized EMD between randomly sampled patches	Small EMD indicates high similarity
Simplicity of a segmentation	Number of basic regions	Fewer regions indicate greater simplicity
	Number of edge pixels	Fewer edge pixels indicate greater simplicity

Table 3.2: Table showing the quantitative measurements used in our evaluation, with their interpretation.

The first two points establish that our segmentations are distinct from but related to the segmentations based on the DOM tree; our algorithm is not just a slower version of these algorithms. The third point suggests advantages in terms of usability; a simpler tree should be easier for users to navigate through or select regions from than a more complex one (*e.g.* one with extraneous edges, per Figure 3.13).

First, our model is based on a principled derivation from intuitively plausible assumptions. Second, our qualitative results indicate that our model places region boundaries in intuitively reasonable positions, and, combined with the quantitative validation, offers evidence to support the usefulness of our model. Table 3.2 summarizes the primary performance metrics that we have used in our validation, when judging the effectiveness of our approach. We note as well that the simplicity offered with our particular approach aligns with key desirable attributes outlined in [75], suggesting that having fewer nodes overall, fewer paths to explore, and structure that minimizes disorientation, is of benefit when users are confronted with hierarchical organization online. In summary, we have shown that our segmentation method has implementation independence, that it produces distinct results from its competitors in practice, that it produces qualitatively plausible segmentations, and that it achieves these desirable properties without sacrificing simplicity to use the image of the rendered page.

3.3 Improved Method

The qualitative tests of the first segmentation algorithm suggested many areas of potential improvement. These improvements have resulted in a new, more sophisticated segmentation method. Like the original algorithm, it proceeds in three stages:

1. Calculation of the probability of a locally significant edge at each pixel
2. Calculation of the probability of a semantically significant line segment between two points
3. Calculation of the most probable segmentation under specific structural assumptions

Each stage has been improved in our new segmentation algorithm, and each affects the others. The method for estimating line probabilities, for example, is determined in part by the requirements of the estimate of segmentation probability, and in turn determines in part the method for estimating the probability of a locally significant edge. Our edge detection method has been improved to use evidence at multiple scales. The calculation of the strength of a line in the current version uses a recursive method to calculate the probability that each segment of the line has a sufficient density of locally significant edges. The calculation of the most probable segmentation now uses an X-Y tree structure originally developed for OCR page segmentation [96]. This tree structure—in which regions are alternately divided horizontally and vertically—and related structures derived from it (*e.g.* [26]), have a long history in optical character recognition (OCR).

3.3.1 Locally Significant Edges

Detecting edges that are relevant to the segmentation is complicated by the presence of very strong edges that make up characters in normal text. These edges are often maximally strong (black on white or vice-versa), but most of these edges are irrelevant to the segmentation. At the same time, faint lines or slight changes in background colour produce very weak edge detector responses, but are often used to show divisions between regions that must be captured by the segmentation. To address this problem, our algorithm estimates the probability that a given pixel contains a locally significant edge. Our edge detection method was designed to meet the requirements of our application (*e.g.* axis-parallel edges and strong responses to faint but locally important edges), but any edge detection method that adapts to local texture and can be used to obtain an edge probability could

be used—the method described by Rakesh *et al.* [106], for example, is a good candidate to be adapted to this problem.

The new algorithm differs from our earlier algorithm in the use of multiscale edge detection with a truncated Gaussian pyramid at the initial stage of Sobel partial derivative operator convolution. The Sobel responses at different scales are added in quadrature to obtain the preliminary edge strengths. This is an improvement since it allows locally significant edges to be found for different neighbourhood sizes and for less-sharp edges.

3.3.2 Semantically Significant Lines

The detection of locally significant edges is not sufficient to define a complete segmentation; these local edges must be "assembled" to produce the outlines of regions. We use an intermediate stage in which the probabilities of extended lines are calculated, and it is from these lines that the final segmentation is built.

The improved segmentation algorithm was designed around the same edge types used for the earlier method and illustrated in Figure 3.2: ridge, step, and alignment edges, plus the borders of the image. Ridge edges are often created by outlines of regions or by the strokes of characters in text. Step edges are often created by changes in background colour used to define regions in a page. Alignment edges consist of sections of edges of the other two types, and commonly appear at the edges of text regions.

The probability that a line is semantically significant is defined recursively. This calculation is shown formally in Algorithm LinePr. For sufficiently long lines (longer than a threshold value t_l), the probability that the line is significant is the probability that both halves of the line are significant. For shorter lines, the probability that the line is significant is the probability that sufficiently many pixels (a proportion greater than or equal to a threshold t_p) along the line have locally significant edges. This prevents broken lines from being assigned an artificially low probability. Given a set of probabilities that individual pixels contain a locally significant edge, the true number n of locally significant edges on the proposed line follows a Poisson binomial distribution (*i.e.* the total number of "successes" in a series of independent non-identically distributed Bernoulli trials). It is possible to calculate this probability exactly (a number of methods are described in [68]), but it has been found to be more efficient in practice to use Monte Carlo simulations to estimate it for the specific distributions, numbers of trials, and numbers of required successes that are encountered by our segmentation algorithm. The use of explicit probability estimation rather than a simple heuristic preserves the Bayesian nature of the method. Note

that, because multiplication is associative, the recursive method is equivalent to taking the product over a sequence of line segments, each with length $\frac{t_l}{2} \leq l \leq t_l$.

Input: Vector L of individual-pixel edge probabilities along the specified line; maximum segment length threshold t_l; minimum edge proportion t_p
Output: Probability P that the line L is present
$l \leftarrow \text{length}(L)$;
if $l > t_l$ **then**
  $P = \text{LinePr}(L_{1 \cdots \lfloor \frac{l}{2} \rfloor}, t_l, t_p) \times \text{LinePr}(L_{\lfloor \frac{l}{2} \rfloor + 1 \cdots l}, t_l, t_p)$;
else
  Let n represent the number of pixels in L that have significant edges;
  $P = \Pr(\lceil \frac{n}{l} \rceil \geq t_p)$;
end
return P
Function LinePr(L, t_l, t_p): Algorithm for calculating the probability of a line.

The recursive definition of the probability of a line has significant advantages over the simple heuristic used in the original segmentation algorithm. The earlier definition stated that the probability of a line was equal to the probability that *each* pixel has a locally significant edge parallel to the proposed line. This does not directly account for the possibility of significant but not continuous edges, such as those formed by alignment; the resulting probability estimates would be higher as the proportion of high-probability local edges increased, but not explicitly accounting for this common case made the probability estimate much less valuable. The difficulty in accurately modelling alignment edges using the earlier model was an important consideration in designing a new line probability model. Additionally, a common difficulty in the earlier method was strong lines tended to result in the algorithm "overshooting", resulting in divisions extending past the region where there was evidence for the line. This is shown in Figures 3.26a and to a lesser extent in 3.26e. Overshooting occurred because the earlier definition of the probability did not account for the distribution of locally significant edges along the length of a proposed line.

The new definition of the probability of the line is a generalization of the one described in Section 3.2.4. When the edge proportion threshold $t_p = 1$, the new method is equivalent to the old; regardless of the segment length, the estimated probability that the line is semantically significant is the product of the probabilities of locally significant edges at each pixel. Similarly, if the segment length threshold $t_l = 1$, $\lceil t_p \times 1 \rceil = 1$ for any $t_p > 0$, resulting in an effective t_p of 1, and equivalence to the older method. We can therefore conclude that both parameters are significant (in that they have an effect on the behaviour of the algorithm), and that the new method is a generalization of the old.

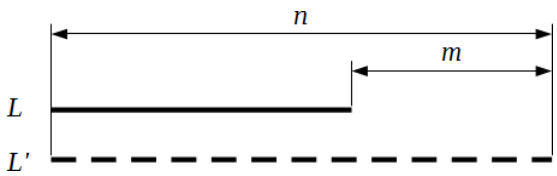


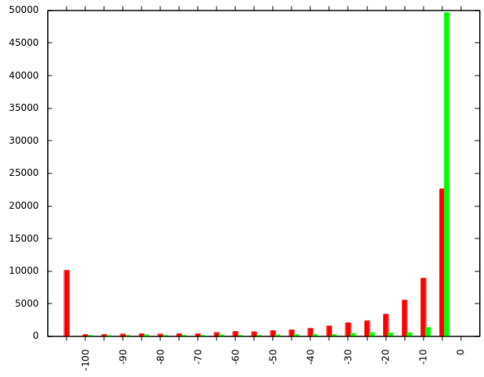
Figure 3.20: Diagram of two proposed lines of length n, each with $n - m$ edge and m non-edge pixels but different semantic significance.

The effects of using a recursive method can be seen by comparing the performance of the two line probability definitions on the proposed lines shown in Figure 3.20. In this example, both proposed lines are the same length, and both have the same number of edge and non-edge pixels, but L only has evidence to support part of its length, while L' has gaps but is clearly semantically significant along its entire length. We have explored the ability of our new method to distinguish between these two cases empirically. Using synthetic example pairs of 2048 pixel partial lines (L) and broken lines (L') with equal overall densities of edge segments, we compared the estimated probabilities for each using many combinations of parameters and local edge probabilities:

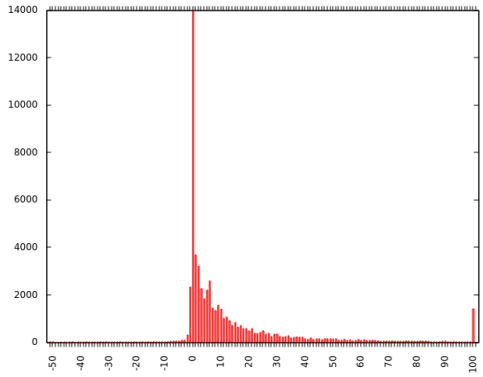
- $t_l = 16i$ for $i \in [1 \cdots 8]$
- $t_p = \frac{1}{i}$ for $i \in [1 \cdots 8]$
- Probability of high-likelihood edges $\alpha = 1 - \frac{1}{i+1}$ for $i \in [1 \cdots 8]$
- Probability of low-likelihood edges $\beta = 1 - \frac{1}{i+1}$ for $i \in [1 \cdots 8]$
- Proportion of high-probability edges $\frac{m-n}{n} = \frac{1}{i}$ for $i \in [1 \cdots 16]$

This results in a total of 2^{16} examples. Overall, $\Pr(L') > \Pr(L)$ for 79.4% of examples, $\Pr(L') = \Pr(L)$ for 14.9% of examples, and $\Pr(L') < \Pr(L)$ for only 5.2% of examples. Figure 3.21 shows histograms of the log likelihoods of L and L' and of the log of the likelihood ratio. The estimated probability of L' is much more likely to be very close to 1 than the estimated probability of L is; in most cases, the log of the likelihood ratio is slightly positive, but high ratios are not uncommon.

Another approach to improving the model of the structure of a line might be to explicitly use the perceptual grouping techniques described by Lowe [88]; we return to this possibility in Section 8.3.2.



(a) Histogram of log likelihoods of semantically significant lines for L and L'



(b) Histogram of log likelihood ratio between L and L'

Figure 3.21: Histograms showing the relationships between estimated probabilities that the lines L and L' are semantically significant.

3.3.3 Segmentations

One important advantage in segmenting images of web pages is that, because of the rendering process, the range of plausible segmentations is much smaller than it is for natural images. We make the same assumptions in defining this segmentation as were used in the earlier segmentation algorithm; in the interest of convenience and clarity we reiterate the assumptions here. The image axes are significant, since they are guaranteed to correspond to the horizontal and vertical axes in the rendering process. Furthermore, regions in a web page are typically represented well by axis-parallel rectangles. We also make the following assumptions:

- Each region is fully contained within its parent region
- Each region is fully covered by its child regions, if any
- No regions overlap unless one region contains the other
- The root of the segmentation tree is a region consisting of the entire page

With these restrictions, every region in a segmentation is tiled by non-overlapping child regions, and all regions are axis-parallel rectangles. While not invariably accurate (some web pages do use overlays or non-rectangular regions), these assumptions allow the use of simple, relatively efficient algorithms to find high-quality segmentations for most pages.

Line probabilities are used to calculate the probability of a division of a region into subregions. In the current algorithm, as in the original, these subregions are assumed to be non-overlapping, axis-parallel rectangular regions that completely cover the parent region. For image segmentation in general, these restrictions would be impractical, but because of the rendering process and design conventions for web pages they are reasonable in practice. The ability to make such strong assumptions about segmentation structure dramatically reduces the search space of segmentations. In the current segmentation algorithm, the possible segmentations are additionally restricted by requiring that all subregion divisions at a given level of the tree be either horizontal or vertical—that is, the segmentation tree is an X-Y tree [96]. This has been found to produce plausible segmentations in practice. In the Bayesian framework of the segmentation algorithm, this corresponds to a prior probability distribution over segmentation trees that is uniform over X-Y trees and zero for all other trees. It also makes the search for the most likely segmentation (in the sense of maximum *a posteriori* likelihood) relatively efficient.

To find a segmentation, we use a recursive algorithm to divide each region, starting from the root region consisting of the entire page. This algorithm finds the highest-probability line across the region, either horizontally or vertically. This is selected as the division line if its estimated probability is at least 0.5, otherwise the region is a leaf region; this procedure is denoted $(N, d) = \text{SplitReg}(n, H, V, s_{min})$, where n is the node to be divided, H and V are maps of locally significant edge probabilities, s_{min} is a minimum size, N is the set of resulting nodes, and d is the direction of division (see Algorithm SplitReg). The complete segmentation algorithm is shown in Algorithm Segment. The segmentation process consists of greedily dividing at the locally optimal points, one division at a time, to produce a set of child regions by either horizontal or vertical division; the regions in this child set are themselves divided until no further "sensible" divisions remain.

3.3.4 Evaluation

We use a number of different methods to evaluate the efficacy of our segmentation, and to compare it to our previous method. We first show a series of examples of the performance of our algorithm, including full pages (Figures 3.23 and 3.24) and a series of excerpts demonstrating specific properties of the algorithm.

Figure 3.22 shows the relationship between estimated edge probabilities and the resulting segmentation for a small section of a web page. Note that the edge probabilities do not correspond exactly to image gradients, but are context-dependent.

Figure 3.23 shows a full page segmented using our algorithm, with divisions outlined in

Input: Node boundaries $n = (n_t, n_b, n_l, n_r)$; horizontal and vertical locally significant edge probabilities H and V; region size threshold s_{min}
Output: Set N of nodes; direction of division D
for i **from** $n_t + s_{min}$ **to** $n_b - s_{min}$ **do**
  $P_{H,i} \leftarrow \text{LinePr}(H(i, n_l \cdots n_r), t_p, t_l)$ (Estimate probability of a horizontal line at row i);
end
for i **from** $n_l + s_{min}$ **to** $n_r - s_{min}$ **do**
  $P_{V,i} \leftarrow \text{LinePr}(H(n_t \cdots n_b, i), t_p, t_l)$ (Estimate probability of a vertical line at column i);
end
if $\max_i(P_{H,i}) \geq \max(P_V) \wedge \max(P_H) \geq 0.5$ **then** Best division is horizontal
  $i \leftarrow \text{argmax}_j(P_{H,j})$;
  $N = \{(n_t, i, n_l, n_r), (i, n_b, n_l, n_r)\}$;
  $d \leftarrow$ HORIZ;
end
else if $\max_i(P_{V,i}) \geq 0.5$ **then** Best division is vertical
  $i \leftarrow \text{argmax}_j(P_{V,j})$;
  $N = \{(n_t, n_b, n_l, i), (n_t, n_b, i, n_r)\}$;
  $d \leftarrow$ VERT;
end
else Region should not be divided
  $N \leftarrow \emptyset$;
  $d \leftarrow$ LEAF;
end

Function SplitReg(n, H, V, s_{min}): Procedure for splitting a region into two sub-regions (if it can be divided). Returns a set N of regions produced by dividing the current region, and the direction d in which the current region was divided.

Input: Node n; horizontal and vertical edge probabilities H and V; minimum size s_{min}
Output: Segmentation tree (N, E)
$N \leftarrow \{n\}$, $E \leftarrow \emptyset$;
$N_{child} \leftarrow \emptyset$;
$(N_{temp}, d) \leftarrow \text{SplitReg}(n, H, V, s_{min})$ (Attempt to divide node);
foreach $n_t \in N_{temp}$ **do** Attempt to divide each node produced
  $(N', d') \leftarrow \text{SplitReg}(n_t, H, V, s_{min})$ **if** $d = d'$ **then** Current division is in the same direction; replace with nodes produced by division
    $N_{temp} \leftarrow (N_{temp} - \{n_t\}) \cup N'$;
  else Current division is in a different direction; add undivided node to child set
    $N_{child} \leftarrow N_{child} \cup \{n_t\}$;
  end
end
foreach $n_c \in N_{child}$ **do** Segment each child region
  $(N'', E'') \leftarrow \text{Segment}(n_c, H, V, s_{min})$;
  $N \leftarrow N \cup N''$;
  $E \leftarrow E \cup E'' \cup \{(n, n_c)\}$;
end
return (N, E)

Function Segment(n, H, V, s_{min}): Algorithm for segmenting a page into an X-Y tree by recursively dividing regions. Divisions are performed one at a time, and all nodes produced by dividing in the same direction are grouped together to maintain the X-Y tree properties. This is equivalent to producing a binary tree by dividing each node that can be divided into two child nodes, then merging intermediate nodes that are divided in the same direction into their parent node.

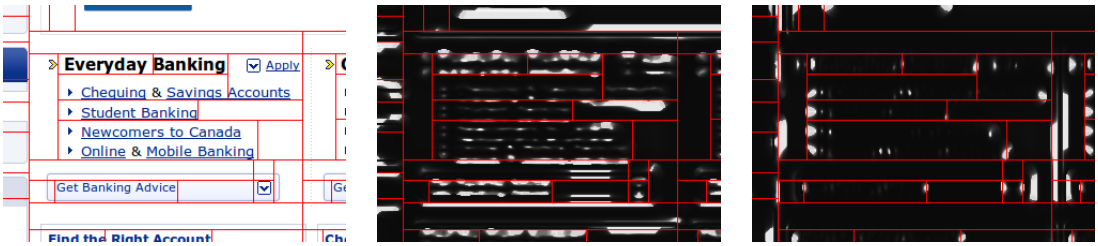


Figure 3.22: Excerpts from `rbc.com` showing, from left to right, the original image, vertical, and horizontal edge probabilities, with the segmentation divisions in red.

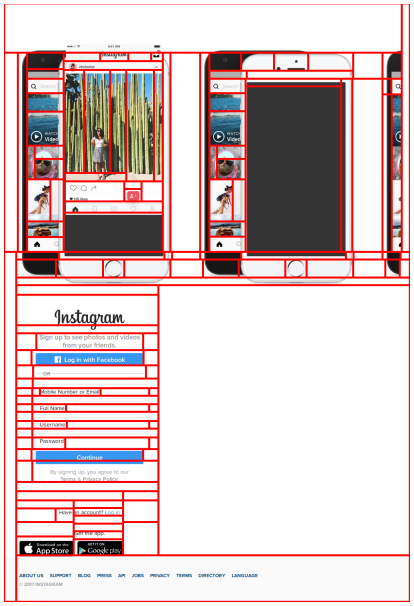


Figure 3.23: Example of a web page segmented using our algorithm.

red. This image was generated using parameters $t_l = 256$ and $t_p = 0.2$. The segmentation shows a plausible structure: major horizontal divisions are detected accurately, as is the division between the form and the region of whitespace. The form is further divided into fields, and whitespace regions are trimmed away. In the illustrations of phones, the thumbnails are accurately separated. There are additional divisions in some of the natural images (*e.g.* between bamboo stalks). Similar divisions have also been observed in other natural images with strong vertical or horizontal edges; a second example can be seen in the background of image in Figure 3.25a. In general, the ability to divide images is an advantage over DOM tree-based methods, since objects such as diagrams or infographics have semantically significant divisions that cannot be expressed in terms of the DOM tree. Future versions of the segmentation algorithm may use statistical properties to distinguish between natural images (for which edge detection may be suppressed) and infographics (for which further segmentation is useful).

Figure 3.24 shows another example of a page segmented by our algorithm, in this case showing the hierarchy of the segmentation tree by colour-coding the divisions. Note that, because the segmentation is required to be an X-Y tree, all divisions at any given level are in the same direction (either horizontal or vertical). Note also the complexity of the segmentation; despite the complex hierarchy of regions and the large total number of regions in the page, our segmentation algorithm has recovered a plausible semantic

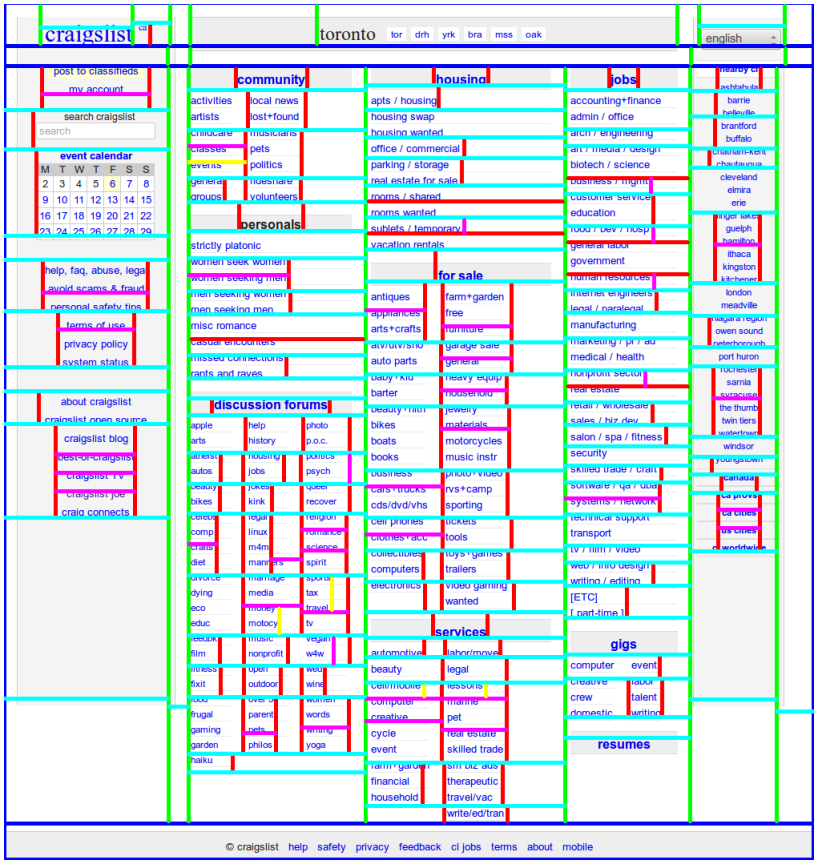


Figure 3.24: Example of a web page segmented using our algorithm. Division colours indicate the level of the division in the hierarchy of the page (with blue at the highest level, followed by green, cyan, red, and magenta).

(a) An overlay results in poor segmentation performance (`www.desjardins.com`)

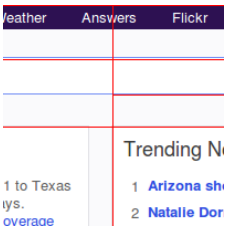
(b) A coincidental alignment is detected in a complex region (`www.cineplex.com`)

Figure 3.25: Examples of regions with properties that degrade segmentation performance

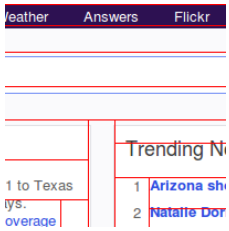
structure.

To demonstrate the improvement our new system represents relative to its predecessor, we show (in Figure 3.26) several examples of cases where the older algorithm failed but the new algorithm succeeds. These examples show the reduced tendency of our new algorithm to "overshoot" as a result of the improved estimate of the probability that a line between two points is semantically significant (Figures 3.26a and 3.26b), much better performance on a form (Figures 3.26c and 3.26d), and improved performance on an overlay on a natural image (Figures 3.26e and 3.26f). In a set of 42 images, we found many similar examples.

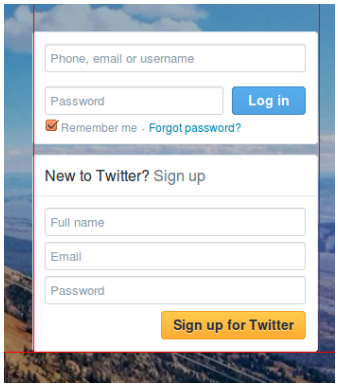
Figure 3.25 shows two examples of particularly difficult regions, with several factors that degrade segmentation quality. First, the use of an overlay in Figure 3.25a conflicts with the tiling model of a segmentation; naturally, the segmentation is more complex than one that explicitly accommodates overlays. There is also a degree of oversegmentation, due in part to coincidental alignments in the very large text. This could be reduced by enlarging the neighbourhoods used in finding locally significant edges to account for the larger font size, or by reducing the maximum segmentation depth. Figure 3.25b shows an example of a coincidental alignment of edges, which produces a spurious division along the left side of the large letter "I" leading up through the text above. A similar issue can be observed in the interior of the large letter "G". Note, however, the accurate vertical segmentation of the large text. Parameter selection could be informed by a study of the statistical properties of region boundaries.



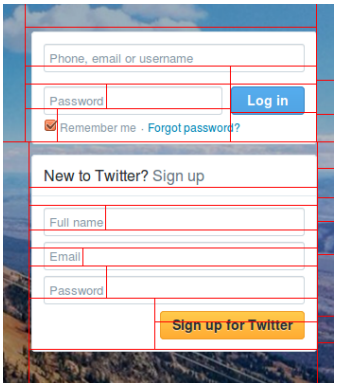
(a) Old algorithm



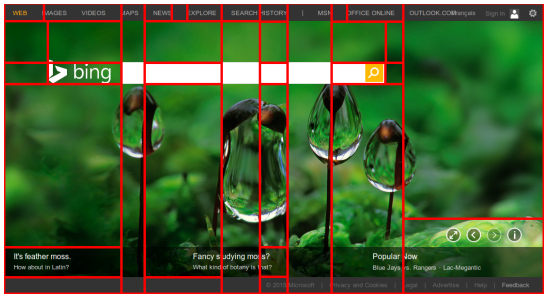
(b) New algorithm



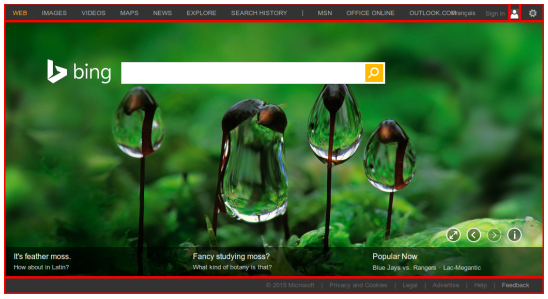
(c) Old algorithm



(d) New algorithm



(e) Old algorithm



(f) New algorithm

Figure 3.26: Comparison of performance between our old algorithm 3.2 and our new algorithm as described here. Examples are taken from `www.yahoo.com`, `twitter.com`, and `www.bing.com`.

3.4 Applications

A web page segmentation algorithm has important applications in the field of assistive technology. Used alone, this type of algorithm can support several transformations of the page, and the segmentation tree can be used by a region classification algorithm to support still more sophisticated assistive interfaces.

Some users have visual impairments that make it difficult to read web pages without any assistive technology, but are not blind. For many of these users, it may be sufficient to simply present regions at a larger scale. Individuals with certain types of cognitive impairment can also benefit from larger text [107]. Simply enlarging the entire page may obscure the large-scale organization of the page, however. Similarly, accessibility features such as Windows Magnifier or the Picture-In-Picture Zoom on Mac OS X, which use a fixed magnification boundary, may not allow the user to read an entire visually coherent region at once. Explicitly segmenting the page into visually coherent regions allows the size of the magnification window to adapt to the page content. A region in the segmentation can be magnified and displayed as an overlay on the page or in a separate window or panel. Allowing the user to reposition the overlay would assist in maintaining context in the page, as the users could uncover nearby regions at will to select the next region they would like to enlarge. Another avenue to explore would be to combine this approach with a "fisheye" distortion around the enlarged areas (as discussed by Furnas [50]) so that the enlarged area does not obscure any of the content; this, however, may prove confusing or distracting for some users. The details of the user-facing front end are a complex challenge in designing assistive interfaces; our focus in this section is necessarily the capabilities that the back-end segmentation system can support, rather than the details of how the transformed page would be presented to the user.

Using a vision-based page segmentation algorithm to simplify a page and eliminate clutter has the potential to benefit many users. Individuals with attention deficits, including age-related declines in divided attention, can find it difficult to filter out distractions and to maintain focus. Such individuals can benefit from adaptations such as highlighting or content isolation to help them to maintain activation on their area of focus [46]. Visuospatial deficits can make it more difficult for users to process and remember the spatial relationships among elements of a page [46], and users with these deficits can benefit from adaptations that reinforce the layout and relationships among items on the page. As aging is associated with a diminished ability to learn and retain new information, and some individuals may benefit from tools that provide cues on the structure or organization of the page [83]. Page segmentation can be used to identify the spatial relationships between regions for appropriate highlighting or simplification.

Some work has also suggested that older adults browse differently from their younger counterparts [47], although this difference appears to be due more to age-associated differences in fluid intelligence than age itself [122]. Eye tracking research suggests older adults attend longer to areas of the page than do younger users [123]. This is perhaps advantageous if that time is spent in considering important content [57], but it can be a disadvantage if it causes more time to be wasted on unimportant or irrelevant elements of the page. An assistive interface that uses page segmentation to isolate visually and semantically coherent regions could be used to display only a subset of these regions at any given time, or to highlight a subset of regions relative to distractors in the page.

Chapter 5 describes a series of experiments aimed at evaluating the ability of our page segmentation algorithm to support an assistive interface. The interface tested includes both magnification and decluttering (implemented by highlighting the focus region). The experiments described include a user study with older adults, and offline tests of the capabilities of the interface. The latter tests allow a more thorough exploration of the performance of the combination of the back-end segmentation system and the front-end assistive interface than is feasible in a user study of reasonable size.

Thus far the focus of this discussion has been on assistive interfaces, but page segmentation may also have applications for users with no assistive needs, especially with respect to convenience features. Content-aware magnification may be useful in many circumstances, especially when a page includes photographs or diagrams that do not match the rest of the text in scale. Highlighting regions may be useful for reducing distractions even for users with no attention deficits, especially on pages that use links with very high visual saliency (*e.g.*, promoted links to other articles on the same site or affiliated sites). Although not essential for web browsing, such convenience features have the potential to make the browsing experience more enjoyable and reduce the annoyances caused by poor design practices.

The algorithms for web page segmentation presented in this chapter were designed with the following key motivation in mind: to provide a principled approach for parsing this class of visual images. We realize that in order to deliver improved experiences for users online, it will be important to also explicitly integrate into our solutions any additional information that may be available (such as certain implementation details for the webpage), and it may also be valuable to deliver support for specific users that is personalized to their particular needs. We discuss the importance of incorporating these components during our discussion of Future Work, in Sections 8.1, 8.3.1, 8.3.2, and 8.3.5. Indeed, working in consort with experts in assistive technology and with input from that user community is an ongoing interest of ours [31, 35] that will also drive new directions with our research.

Our initial motivation for exploring this class of images in order to enable improvements for the assistive needs community came from our study of the current state of screen readers [32]. As we began joint research with an HCI expert who had particular interest in the WebforAll (W4A) community [35], we were made aware of the possible value of our parsing of web pages as visual images for the front end tasks of decluttering and zooming. Positive reactions from mentors in the W4A community [31] provided additional reinforcement of our particular approach as a valued one.

Chapter 4

Region Classification

In this chapter we present a method for classifying regions in a web page using purely vision-based methods. Our algorithm employs the novel approach of using a hidden Markov tree (HMT) to represent the structure of a segmentation tree; the classification process, like the segmentation process, is entirely Bayesian. The classification framework used in this research is described in Section 4.2. Section 4.3 describes the results of the first series of experiments carried out using this approach, and Section 4.4 describes a new set of experiments designed to address issues with the ontology and labelling process uncovered by our first set of experiments. In Section 4.5, we discuss the possible applications of region classification.

4.1 Motivation

From the perspective of assistive technology, classification of web page regions is valuable because it allows an assistive interface to interact with and modify the page in more sophisticated ways and at a higher semantic level. This is especially important for screen readers. As described in Chapter 2, complex web pages are often very difficult to navigate using screen readers. Semantic labels of regions, such as ARIA labels [128], can help users navigate through areas of the page by providing high-level semantic information about the role of each region in the page. Unfortunately these labels are often not present, and this can force the user to navigate through the DOM tree based on HTML tags and similar implementation-level evidence. This leads to a complex navigation problem, and may be intimidating for users who are unfamiliar with HTML. The automatic classification of web

page regions into clear, semantically relevant categories such as those provided by ARIA labels or the EMine ontology [3] can help to address these problems.

A tree of labelled regions will have many other applications outside of screen readers, and even outside of assistive technology. Interfaces designed to prune away specific types of content could be used to reduce distractions, in a more sophisticated version of the segmentation-based decluttering interfaces described in Chapters 3 and 5. Such interfaces could also be used to modify content for more appealing presentation (*e.g.*, by removing ads), for other devices (*e.g.*, on the small screens of mobile devices), or to automatically extract relevant content (*e.g.*, for generating a printable version of a page). The high-level labels provided by a classification algorithm, when coupled with a segmentation tree, allow very high-level transformations of web pages for a wide variety of purposes.

From the perspective of computer vision, and the problem of object recognition in general, web pages provide an interesting domain and an interesting set of challenges. As pointed out in Chapter 3, web pages have very complex structures with very large numbers of regions. Intuitively, it is easy to see that context is very important in web page classification; a list of links in the footer of the page serves a different purpose from a list of links in the header, and an image in the header is probably some sort of logo, while an image in an article is probably an illustration. Both of these factors make the problem challenging. The powerful assumptions that can be made about region structure and the presence of a natural scale in the image also allow some problems in object recognition to be abstracted away when designing features, which makes the domain a very useful one for exploring techniques in a complex, highly contextual setting. For example, in natural images (and other images formed by perspective projection), the scale of an object in the image depends on its inherent size, distance from the camera, and the camera parameters, producing examples with a continuous range of scaling factors for examples of a class. Web page images, however, have a natural scale and are free of perspective effects, leaving only the inherent size of regions variable.

4.2 Classification Algorithm

We view the problem of classifying regions in the image of a rendered web page to be the problem of inferring a set of region labels from the observed properties of each region and their relationships to other regions. This implies four sub-problems:

1. Obtaining a set of regions from the image of a rendered web page

2. Observing the relationships between regions
3. Observing the properties or features of individual regions
4. Inferring the most likely (in the sense of maximum *a posteriori* probability) set of labels for the set of regions as a whole

4.2.1 Segmentation

We address the first two sub-problems by assuming that our classification method takes as input not the original page image, but a hierarchical segmentation of the page image. Classification will then proceed, by design, independently of the segmentation algorithm. (Other approaches perform segmentation and classification simultaneously; we discuss these in Section 7.1.) It is, however, ideal to use a segmentation algorithm designed specifically for segmenting web page images, such as one of the algorithms introduced in Chapter 3. Briefly, our segmentation algorithms work as follows:

1. The algorithm starts with a "root" region corresponding to the entire page.
2. The region is searched for horizontal and vertical edges that "stand out" from nearby edges.
3. "Candidate boundaries" are found where many of the strong edges found in step 2 are aligned.
4. These candidate boundaries are used to produce a tiling of non-overlapping subregions completely covering their parent region, with every edge in the tiling corresponding to a candidate boundary (although not every candidate boundary corresponds to an edge in the tiling). The tiling is selected to have the best support from the evidence in the image, subject to a maximum number of child regions (*i.e.*, a greedy search at each level).
5. Steps 2 through 4 are repeated for each child region (if multiple child regions are found), until a maximum tree depth is reached.

This method produces a tree of regions in which the root corresponds to the entire page, and leaves correspond to regions at the maximum depth or with insufficient evidence to support further division. Every part of the page is contained in some leaf node, all descendants of a given region are completely contained within it, and sibling nodes do not overlap.

4.2.2 Feature Extraction

The third sub-problem is the problem of defining a feature vector to represent each region. We prioritize simplicity and rapid calculation in selecting features. Intuitively, it is reasonable to expect regions of specific types will be found in specific areas of the page; the main content, for example, is likely to be found near the center of the page, and a sidebar is by definition found on the side of the page. Similarly, the size and proportions of regions are likely to be informative. Finally, the types of content found in a region are likely to prove important. We use the following region features:

- Position of the region origin relative to the entire page and parent region, normalized such that the width of the page or parent region is 1 (4 dimensions)
- Area relative to the entire page and parent region (2 dimensions)
- Logarithm of aspect ratio
- Proportion of text in the region
- Proportion of images in the region
- Proportion of other content in the region

These features are summarized in a ten-dimensional real-valued feature vector for each region. Other features could also be chosen, but these were found to be effective in the experiments described here.

We use the logarithm of the aspect ratio to describe the proportions of the rectangular regions. Using the aspect ratio itself is initially appealing, but a problem arises in comparing aspect ratios. Consider a square region and a region with one dimension ten times as long as the other. The aspect ratio of the square region is 1; if the elongated region's long edge is horizontal, its aspect ratio is 10, and if it is vertical its aspect ratio is 0.1. Thus, using the simple aspect ratio, the distance between the two regions is 0.9 in one orientation and 9 in the other. Using the logarithm (in base 10, for this example) of the aspect ratio, the square region has a value of 0, and the elongated a value of either -1 or 1; the distance will be the same in each case. In general, the use of the logarithm of the aspect ratio makes the absolute value of differences between two regions invariant with respect to a right-angle rotation of the pair of regions.

Our method of calculating the proportions of text and images in a region uses a preprocessing step to classify $n \times n$-pixel blocks containing text and images in the region. As an

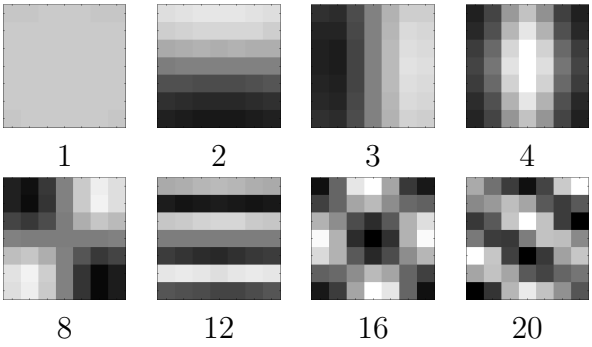


Figure 4.1: "Eigenblocks" and corresponding indices found for representing 7×7 blocks from web page images. These blocks are used to detect text and natural images.

initial step, we use an algorithm analogous to eigenfaces [124] to find a set of "eigenblocks" that comprise a good basis for the distribution of blocks found in web pages (*i.e*, the principal components of the set of blocks, considered as vectors of pixel values), based on a large sample of blocks from a variety of web pages. Figure 4.1 shows a selection of these blocks. We then express blocks in a training set of web pages (disjoint from the set of web pages used for training or testing the classification of regions) in terms of this new basis; we used all components rather than reducing the dimensionality of the problem by using a subset of the eigenblocks, since this was found to help accuracy. Using classifications for these blocks obtained from the DOM tree of the page as training data, we learn a linear classifier for text blocks and a linear classifier for image blocks.

Note that although similar principal component representations to our eigenblocks have been used in studying image statistics (see, *e.g.*, [98] and [56]), our use here is limited to classification of content types. In the future, we are interested in studying the statistics of images of rendered web pages (see Section 8.3.4). In such a study, principal components analysis, as used in [56], or related methods such as independent components analysis [13] can be expected to play a prominent role as they have in the study of the statistics of natural images.

Rather than hand-labelling images and text in the images of web pages, we use DOM tree information to infer the positions of text and images in the image of the page. Image detection simply uses the bounding boxes of <IMG>-tag nodes. Text detection uses the bounding boxes of element nodes that contain only text nodes. Using 7×7 blocks (as described above), a simple linear classifier trained on approximately 5.2×10^6 labelled blocks achieves an accuracy of 96.1% (on a disjoint test set of similar size) for the detection of text, and an accuracy of 98.9% for the detection of images. The block size was selected

with the constraints that it should be small enough to be efficient to train and use, while being large enough to include salient features for distinguishing text from natural images (*e.g.*, the hard, high-contrast edges found in text).

4.2.3 Inference

The fourth and final sub-problem is the problem of determining the jointly most likely set of labels given both the structure of the segmentation of the page and the individual features of each region. This is accomplished by defining a hidden Markov tree or HMT—a generalization of a hidden Markov model from a sequence structure to a tree structure (described in the context of wavelet analysis in [36])—based on the structure of the segmentation tree, with hidden states representing the labels of regions and observations representing the feature vectors describing each region. The HMT is produced from the segmentation tree according to the following rules:

1. For each non-root node in the segmentation tree, there is a corresponding hidden node representing the label of that region in the HMT
2. The root node in the segmentation tree corresponds to an observed node in the HMT representing the known label ("`fullpage`") of the root region
3. For each edge connecting a parent region to a child region in the segmentation tree, there is a corresponding edge in the segmentation tree connecting the nodes representing the regions' labels
4. Each label node in the HMT has an observed child node representing the feature vector describing the corresponding region in the segmentation tree

Figure 4.2 shows an example of such a transformation for a simple segmentation tree. It is worth noting that the HMT does not require an X-Y tree structure, only a tree structure of some sort. As a result, modifying the assumptions made in the segmentation process (e.g. by relaxing the X-Y tree prior to allow for overlays) will not invalidate the HMT method.

To determine the most probable labels, we use the well-established belief propagation algorithm [100]. Briefly, this algorithm represents the belief distribution about the value of each node as the product of two factors, π and λ, which are updated by messages passed

from parent to child (in the case of the π distribution) or from child to parent (in the case of the λ distribution). The π-message passed from a parent p to a child c_i is defined to be

$$\pi_{p,c_i} = \alpha\pi(p)\prod_{j\neq i}\lambda_{c_j,p},$$

where c_j is also a child of p and α is a normalizing constant. The message sent from a child c to its parent p is defined to be

$$\lambda_{c,p} = \sum_{\text{Values of } c}\lambda(c)M_{P(c|p)},$$

where $M_{P(c|p)}$ is a matrix representing the probability distribution of values of $P(c|p)$, such that the value at row c and column p is $P(c|p)$ (for some indexing function converting labels to row and column indices). The π-factor for each node i is updated according to the equation $\pi(i) = M_{P(i|p)}\pi_{i,c}$, and the λ-factor according to the equation $\lambda(i) = \prod_{c_j}\lambda_{c_j,i}$, where p is the parent of i and each c_j is a child of i. Observed nodes have λ equal to zero except for the observed state (set to 1), and unobserved leaf nodes are initialized with a uniform λ. At each iteration, belief updating, top-down propagation, and bottom-up propagation can be performed for each node in any order; iteration proceeds until convergence. Finally, the belief $Bel(i) = \alpha\pi(i)\lambda(i)$, where α is a normalizing constant.

In our implementation, nodes are added to the tree starting from the root in an order that ensures that all calculations have sufficient information to complete (*i.e.* all nodes have an initial π and λ from a previous iteration, are observed, or are unobserved leaf nodes)[1]. The process is iterated to convergence with each set of nodes before new nodes are added. Our implementation is derived from a detailed description of belief propagation on tree-structured Bayesian networks in [100].

The large-scale structure of the HMT varies for each image with the structure of the segmentation tree, but the small-scale structure is consistent. A hidden label node has one parent (a label node corresponding to the label of the parent region), one observed child node (representing the feature vector describing the region), and possibly one or more hidden child nodes (representing the labels of child regions). This local structure is shown in Figure 4.3. To calculate the messages to be passed by node i, the following probability distributions must be calculated:

- $P(Label(i) = l_1|Label(Par(i)) = l_2)$

[1]That is, inference occurs as the HMT is constructed, in order to ensure that sufficient information is available for message passing from each node.

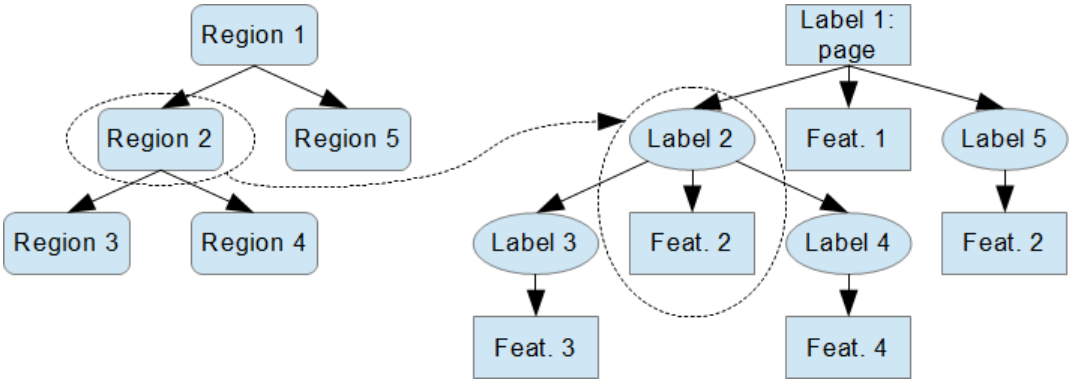


Figure 4.2: Example of a segmentation tree (left) and the corresponding HMT (right). For each region in the segmentation tree (shown by rounded rectangles), there is a corresponding label and observed feature vector in the HMT (one such transformation is shown in dotted lines). All feature vectors are observed; all labels are hidden save for the label of the root region, which is known to be the full page. In the HMT, ellipses represent hidden nodes and rectangles represent observed nodes.

- $P(Feat(i) = \mathbf{x}|Label(i) = l)$

In these equations, $Par(i)$ represents the parent node of i, $Label(i)$ represents the label assigned to the region i, and $Feat(i)$ represents the feature vector describing region i. Under the assumption that the same conditional probability distributions apply to every label node in the tree, the same set of distributions suffices to calculate all messages passed during the execution of the belief propagation algorithm, despite the varying large-scale structure of the HMT.

These conditional distributions are defined by empirical distributions learned from the class labels (in our first experiment, ARIA role labels) in the dataset of web pages. For the discrete distributions over child labels given the parent's label, this simply consists of counting occurrences of label pairs that occur in a set of web pages used as training data. We define this probability distribution to be

$$Pr(Label(i) = l_1|Label((Par(i)) = l_2) = \frac{A}{B} \tag{4.1}$$

where

$$A = |\{j \in T : Label(j) = l_1 \wedge Label(Par(j)) = l_2\}| + \epsilon \tag{4.2}$$

and

$$B = |\{j \in T : Label(Par(j)) = l_2\}| + k\epsilon \tag{4.3}$$

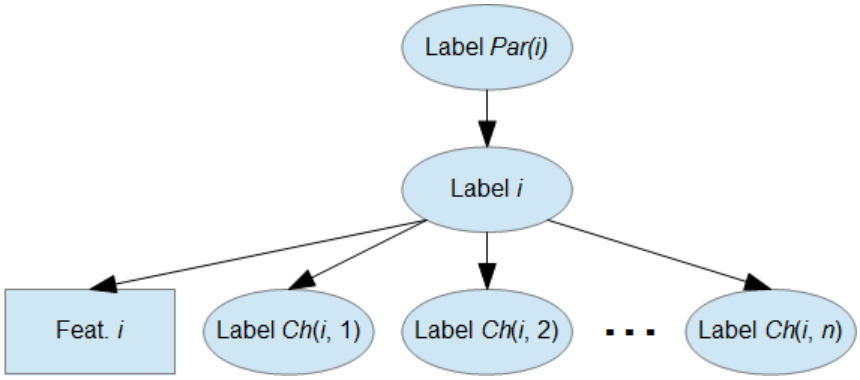


Figure 4.3: Local structure of the HMT used for classification. In this diagram, $Par(i)$ represents the parent of region i, and $Ch(i, j)$ represents the j^{th} child of region i.

In these equations, T represents the hand-labelled training set, k represents the number of possible labels, and ϵ is a small constant used to ensure that combinations of parent and child not found in the training set can be predicted given strong enough evidence. The probability distribution of feature vectors given a semantic region label is more complex, since the feature vectors are continuous. To represent these distributions, we use kernel density estimation with a Gaussian kernel. Thus, for any feature vector $\mathbf{x}$ describing a region i,

$$Pr(Feat(i) = \mathbf{x} | Label(i) = l) = \sum_{j \in T_l} \left(\frac{Norm(\mathbf{x}; Feat(j), \sigma)}{|T_l|} \right).$$

In this equation, T_l represents the set of regions in the training set with the label l, and $Norm(\mathbf{x}; \mu, \sigma)$ represents the probability density at position $\mathbf{x}$ for a Gaussian with mean μ and standard deviation σ; our experiments were performed with $\sigma = 0.005$.

Summary

The value of our approach to classification may best be seen in the following way. We begin with a segmentation tree representation of a webpage, which shows regions, but not labels. We progressively learn what these labels should be, on the basis of feature vectors. But we do not simply predict each region label separately. Instead, we take into consideration the parent-child relationships from the segmentation tree and think in terms of the relevant context within the page (since depending on that context, the labels will truly differ).

In essence, for our classification solution, the context and the actual learned probability distribution are considered together within the same graphical model.

4.3 Early Results

To demonstrate the efficacy of our classification method, we compare the classification results with ground-truth data derived from ARIA landmark role labels. These labels are designed for web accessibility, but are optional, and many web pages do not make use of them. Our dataset includes the top 35 pages in Canada (according to Alexa[2]) which do use ARIA roles[3]. This is a diverse set of pages; the content includes search, news, shopping, and social network web sites, and the visual designs range from Google's minimalist home page to the busy multi-column layout of the CBC's home page. Other, similarly-popular pages without ARIA role labels were used to train the text and image classifiers for feature extraction. Although the dataset contains relatively few *pages*, each page contains many individually-labelled *regions*; at a region level, there are 4361 items in the dataset. We use the full set of ARIA landmark role labels:

1. `application`
2. `article`
3. `banner`
4. `complementary`
5. `contentinfo`
6. `directory`
7. `document`
8. `form`
9. `log`
10. `main`
11. `navigation`
12. `note`
13. `region`
14. `search`
15. `status`

Additionally, we use several ARIA widget role labels:

16. `button`
17. `grid`
18. `gridcell`
19. `heading`
20. `img`
21. `separator`

Of these labels, only ten occur in our dataset. By their index in the list above, these labels are 3, 4, 5, 10, 11, 14, 15, and 16. We augment this set of labels with two others:

[2] `alexa.com`

[3] Of the top 50 pages in Canada, only these 35, or 70%, use ARIA labels; since they are not universal even among the most prominent web sites, it is clear that they cannot be exclusively relied upon on the Web as a whole.

Label	Count	Label	Count
`fullpage`	35	`navigation`	148
`banner`	138	`search`	31
`complementary`	150	`status`	1
`contentinfo`	80	`button`	9
`main`	1161	`unlabelled`	2608

Table 4.1: Frequency of occurrence of each class in the dataset

`fullpage` (index 0), representing the complete web page at the root of the segmentation tree, and `unlabelled` (index 22), for regions without an ARIA role label. Only these labels are used in our experiments. The numbers of occurrences of each label are shown in Table 4.1. Note that the classes are not balanced; this makes accurate classification more difficult.

Each page in the dataset was segmented (as described in Section 3.2, running the algorithm shown in Appendix A, as this experiment was performed before the improved algorithm was available), and the resulting regions were matched with ARIA roles. Since our segmentations do not necessarily correspond exactly to the bounding boxes of the DOM tree elements labelled with the ARIA roles, we label each region in the segmentation according to the predominant non-empty label of the region. When more than one label applies to a region (*e.g.* a `gridcell` inside a `grid`), the label of a descendent overrides the label of an ancestor. This process is illustrated in Figure 4.4.

There are several potential sources of noise in producing our training and testing data. ARIA labels can only be applied by the page designer to whole DOM tree nodes, and cannot distinguish between regions of black-box objects such as areas within images. The translation between the DOM tree nodes and the nodes of our segmentation tree, as shown in Figure 4.4, may also occasionally introduce noise where the two segmentations are not aligned. To overcome these potential problems, we use a training set with a large number of nodes (a total of 4361); because ARIA role labels are included in the page source code, a large dataset of regions can be built more quickly than by using manual labelling.

To make optimal use of the dataset, we use leave-one-out cross-validation. Thus, in our set of 35 web pages, we run 35 rounds in which we train the algorithm on 34 pages and test on the remaining page.

Table 4.2 shows the confusion matrix for role labels. This shows not only overall accuracy, but the frequency with which the classification algorithm confuses each pair of

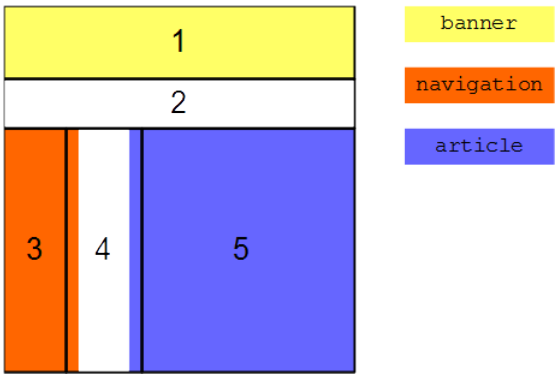


Figure 4.4: Association between labels in the DOM tree (solid colours) and regions in the segmentation tree (bold outlines). Using the predominant DOM tree node label for each segmentation tree region, regions 2 and 4 have no label, region 1 has the label "`banner`", region 3 has the label "`navigation`", and region 5 has the label "`article`".

	0	**3**	**4**	**5**	**10**	**11**	**14**	**15**	**16**	**22**	**Total**
0	35	0	0	0	0	0	0	0	0	0	35
3	41	12	2	0	3	11	12	22	19	16	138
4	2	0	6	3	5	1	13	0	0	120	150
5	0	0	3	32	1	2	0	0	0	42	80
10	15	5	46	20	545	8	46	3	4	469	1161
11	43	11	2	1	7	20	21	3	24	16	148
14	2	2	0	0	1	14	8	0	1	3	31
15	0	0	0	0	1	0	0	0	0	0	1
16	1	0	0	0	0	0	0	0	8	0	9
22	102	49	163	114	462	57	138	27	123	1373	2608
Total	241	79	222	170	1025	113	238	55	179	2039	4361

Table 4.2: Confusion matrix showing ground truth labels in rows and predicted labels in columns. Each cell shows the number of occurrences of that pair of true and predicted labels. The indices of each label are taken from the list of role labels above.

Class	0	3	4	5	10	11	14	15	16	22
TP	35	12	6	32	545	20	8	0	8	1373
FP	206	67	216	138	480	93	230	55	171	666
TN	4120	4156	3995	4143	2720	4120	4100	4305	4181	1087
FN	0	126	144	48	616	128	23	1	1	1235
Precision	0.15	0.15	0.03	0.19	0.53	0.18	0.03	0.00	0.04	0.67
Recall	1.00	0.09	0.04	0.40	0.47	0.14	0.26	0.00	0.89	0.53
Accuracy	0.95	0.96	0.92	0.96	0.75	0.95	0.94	0.99	0.96	0.56

Table 4.3: Table showing several performance measurements for each class.

labels. In the confusion matrix, M, M_{rc} is the total number of times a region of class r was classified as a region of class c. Diagonal elements represent correct classifications, and off-diagonal elements represent incorrect classifications. The overall accuracy of 46.8% is far better than the chance accuracy of 10% for a ten-class classification problem. Table 4.3 shows performance measurements for each class, including true positives (TP), false positives (FP), true negatives (TN), false negatives (FN), precision ($\frac{TP}{TP+FP}$), recall ($\frac{TP}{TP+FN}$), and overall accuracy ($\frac{TP+TN}{TP+FP+TN+FN}$). It is clear that performance for the dominant classes (in terms of the number of examples) is good, although performance for rarely-encountered classes suffers by comparison.

There are a number of ARIA roles that distinguish on semantic grounds types of content that are likely to be visually similar. The labels `main` (the main content of the page, index 10) and `complementary` (content that is complementary to the main content, index 4), for example, both refer to content rather than navigation or interactive widgets, and are likely to be similar in appearance since they differ primarily in the relationships of the meaning of the content. If these are combined, the resulting class is classified with a precision of 0.48 and a recall of 0.46, both slightly less than the larger class `main` alone, but a dramatic improvement on the values for the smaller class `complementary`.

The most common errors are classification of a labelled region as `unlabelled` (index 22) and classification of an `unlabelled` region with a role label. The distribution of ground truth labels in the dataset suggests that this may be due to incomplete labelling of many web pages with role labels when they are present. While there are no `img` labels in the dataset, for example, there are many images in the web pages. For regions where neither the ground truth label nor the predicted label is `unlabelled`, the predicted label matches the ground truth label exactly in 61.3% of cases.

Because the ARIA labels are originally associated with DOM tree nodes, and our visual

segmentation's boundaries are not necessarily identical to the boundaries of the DOM tree nodes, the association of labels with nodes in our segmentation tree may introduce noise in the training data. To check for this, we also test the classification algorithm using a segmentation generated using the bounding boxes of DOM tree nodes as the boundaries of segments. Essentially, this is equivalent to a hand-labelled segmentation produced by the page designers. Although this segmentation is not generated from purely visual data, it does eliminate a source of noise in the training data, and the classification is done using the same visual features. For our tests, we use 29 pages, omitting 6 for which the DOM tree structure was not appropriate for processing and feature extraction. Because of the prevalence of unlabelled nodes in the DOM tree, we set the label of each node whose children all share the same label to that shared label; even with this measure, however, 29857 of 30210 DOM tree nodes (98.8%) are still unlabelled, much more than in our visual segmentation. The overall accuracy is lower at 37.9%. Most of the errors are due to unlabelled regions being assigned a label; like the similar errors produced when classifying regions in the visual segmentation, this may simply be due to the page designers not assigning labels to regions where they would be appropriate. Considering only regions with ground-truth or predicted classes of `unlabelled`, the accuracy is slightly lower still at 35.3%. These results do not indicate that the process of associating DOM tree ARIA labels with regions in the visual segmentation is generating problematic noise in the training data.

Unlabelled regions are a problem which could be approached in several ways. One approach would be to examine methods for distinguishing regions which are unlabelled because the page designer neglected to assign a label from regions to which no label is applicable (*i.e.* to distinguish between training examples that have no label attached, and training examples with an implicit label of "other", such as a whitespace region). With this distinction, it would be possible to use a semi-supervised learning method to estimate the probability distributions to be learned, or simply prune the training set, allowing us to overcome the problem of unlabelled regions. This approach has some appealing aspects, but is labour-intensive to implement and may require a very large dataset to learn to distinguish between different types of unlabelled regions. Instead, we turn to manual labelling. Although also somewhat labour-intensive, manual labels can be assumed to reflect the beliefs of the labeller about the ground truth labels of regions, and as a result manual labelling avoids the possibility of a semi-supervised learning method producing incorrect results due to an insufficient dataset or incorrect assumptions about the distributions of labelled and unlabelled regions of different classes.

4.4 Improved Experiments

The ARIA labels used in the first experiment are not always sufficient to label all regions in a page; some regions were not accurately described by any ARIA label and were therefore unlabelled. There were also many regions for which an applicable label was not assigned by the page developers. These two issues combined to produce a noisy dataset with large numbers of unlabelled regions for use in training and testing; it is therefore not surprising that the classification algorithm struggled to achieve high performance in these experiments. In the experiments described in this section, we have worked to address these problems: the issue of regions not having an applicable label has been addressed by using a new ontology; the issue of incomplete labelling by page developers has been addressed by using manually-labelled pages. By the time these experiments were conducted, the improved segmentation algorithm described in Section 3.3 was available; we used this algorithm in these experiments.

4.4.1 Ontology Selection

Ontology selection is, in general, a very difficult problem. It is important, when classifying regions, to select a set of labels that is both comparable in granularity to the segmentation and relevant to the task at hand. For an extreme case of granularity mismatch, consider an algorithm that produces a high-quality segmentation at the level of large regions (such as articles and headers), and a set of labels corresponding to individual characters. Regardless of the algorithm used to assign labels to regions, the results will inevitably be uninformative. The same set of labels could, however, be used effectively with a character-level segmentation. A page that is over-segmented with respect to one ontology may be under-segmented with respect to another and perfectly segmented with respect to a third. Relevance is equally important, and depends to a large extent upon the intended application. Continuing with the example, character-level labels are highly relevant to optical character recognition (OCR), but are almost completely irrelevant to the problem of rearranging major page elements to suit a small screen. In the domain of assistive technology this is an especially complex problem [128].

Work by Akpinar and Yesilada [3] presents an ontology-based heuristic approach for identifying visual elements of web pages and their roles, implemented by extending and modernizing the well-known VIPS algorithm [22]. Like VIPS and its many other descendants, the approach described by Akpinar and Yesilada relies upon the DOM tree. The objectives of this algorithm were to improve accessibility and the presentation of pages

on small screens (*e.g.*, on a cell phone). Their algorithm is, however, designed to assign roles to visual elements using a probabilistic approach, as ours is. An online survey was conducted for user evaluation and 80% of the users were receptive to the interpretations produced, although some differences in user points of view were also noted [3]. We employ the set of classes selected by Akpinar and Yesilada for the eMine ontology [3], on the basis that it is designed to be relevant to assistive technology and experiments with users have confirmed that it can provide plausible descriptions of web pages. The use of an existing ontology also helps to avoid creating bias by selecting labels that play to the strengths of our classification algorithm.

Akpinar and Yesilada's eMine ontology is a descendent of the WAfA ontology [137]. In some respects eMine is coarser-grained, in that some roles with no clear distinguishing features are combined; in other respects, eMine extends the WAfA ontology by providing additional roles. Additional roles were selected based upon the ontologies underlying ARIA labels [129] and HTML5 roles [17]. This effort to move beyond ARIA labels is well suited for our purposes; it resonates with discussions we had at the 2016 Web4All conference [31] with assistive technology experts, some of whom had assistive needs themselves. The observation was that ARIA labels alone are not sufficient to support screen readers, but were intended to supplement tag information. A more comprehensive set of classes represents an important advantage from the perspective of supporting screen reader navigation. The complete eMine ontology includes not only a set of classes but also a set of heuristic rules defining their characteristics.

Our dataset includes regions with the following labels from classes in Akpinar and Yesilada's eMine ontology:

- `Page`
- `Article`
- `Body`
- `BreadCrumbTrail`
- `Caption`
- `ComplementaryContent`
- `Container`
- `Copyright`
- `Figure`
- `Footer`
- `Header`
- `Icon`
- `Link`
- `List`
- `Logo`
- `Margin`
- `Menu`
- `MenuItem`
- `Nothing`
- `Separator`
- `Sidebar`
- `SpecialGraphic`
- `Title`
- `TitleBanner`

To this set we added an `Empty` class for whitespace segments detected by our segmentation algorithm. We only included roles in our classification experiments if they were found in our dataset at least once. These roles are similar in semantic level to the ARIA labels and HTML tags commonly used for screen reader navigation today.

4.4.2 Manual Labelling

The second key difference between the experiments described here and those described in Section 4.3 is that the ground truth used for training and evaluation is manually (rather than automatically) generated. In order to manually classify a large number of regions, we developed a simple graphical interface to allow a human classifier to select and label regions in the page. The regions were labelled one level of the tree at a time; since regions at the same level cannot overlap in the segmentations produced by the algorithm used here, this simplifies the interface by removing ambiguity in selection. In order to efficiently label areas where many region shared the same label (a common occurrence, since web pages often contain many instances of the same type of content, such as article blurbs in a news site home page or image thumbnails in a gallery page), the interface included the ability to simultaneously select multiple regions and apply the same label to all. Figure 4.5 shows a screenshot of the interface in use.

The principal difficulty encountered in manual segmentation was the scale of the task. The segmentation tree of a web page often has 50-60 layers of regions, and in each layer there could be hundreds of regions to be labelled; as a result, the labelling process was labour-intensive. Oversegmented and undersegmented regions in the segmentation tree were also challenging to label accurately and descriptively.

To ensure consistency in labelling, a single researcher labelled all pages in the dataset. As a result, the model learned reflects the perception of one researcher. With a larger dataset, it may be feasible to investigate the differences in perception between different users. If agreement between labellers is high, this suggests that labels can be confidently assigned to regions and that evaluation with a single ground truth classification should provide accurate results. If opinions on the correct labels differ between users, a single ground truth classification will not capture the range of perceived classifications, and a "one-size-fits-all" classification method will not be able to satisfy all users without personalization. Using a single labeller, we do not need to account for between-labeller variations, but we cannot answer the question of consistency. We discuss this issue further in Section 8.3.5. The issue of individual differences in perception will be addressed with respect to ground truth segmentations in Chapter 6.

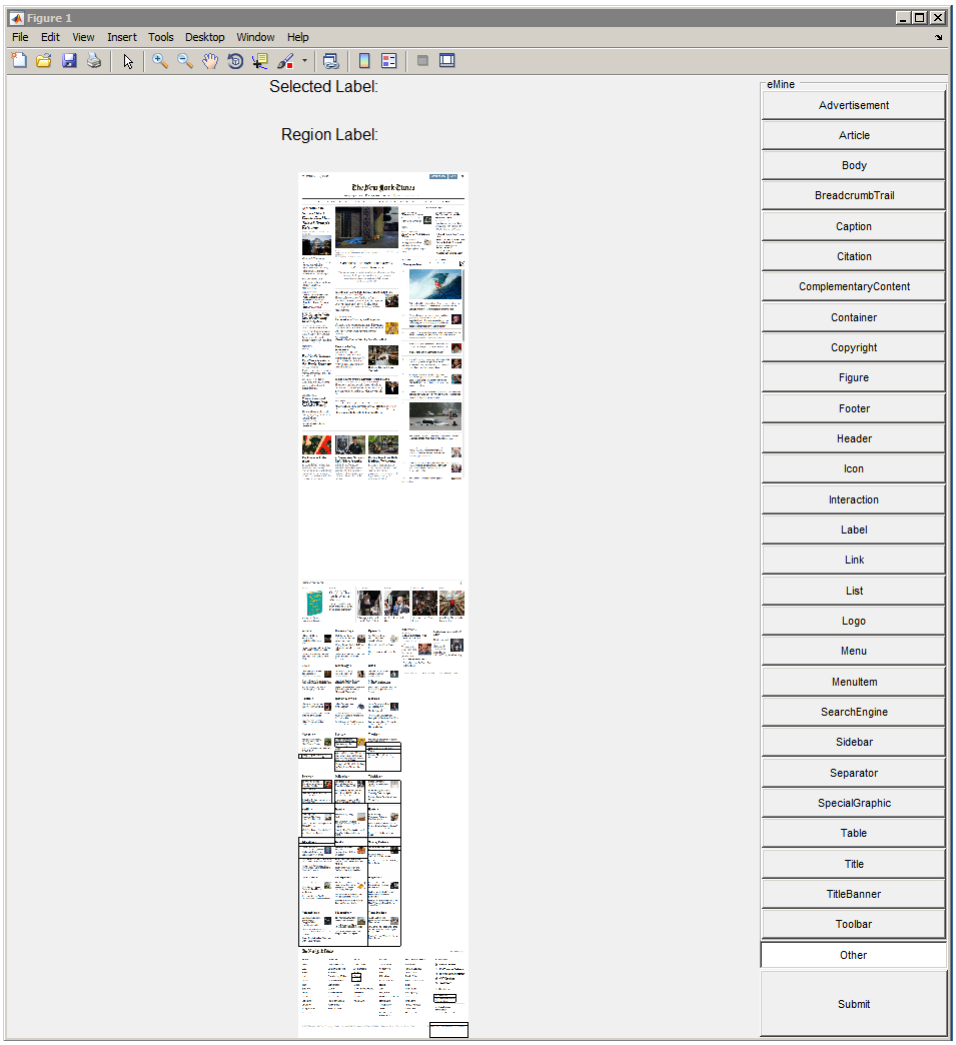


Figure 4.5: Screenshot of region labelling interface, showing page (main area) and list of labels (right sidebar).

Manual labelling not only avoids the problem of missing labels, but also associates labels directly with regions in the segmentation used. In Section 4.3, the ARIA labels were associated with regions in the DOM tree, which were then associated with the regions in the visual segmentation tree. In testing the visual segmentation algorithm, we have already demonstrated that the visual segmentation tree differs substantially from the DOM tree; as a result, the process of automatically associating ground truth labels with regions is another potential source of noise in the training and evaluation data used in the original classification experiments.

4.4.3 Dataset

For this series of experiments, we have selected a set of nine news web pages: three each from BBC News, CBC News, and the New York Times. News pages are useful for this type of research because they are very complex, with a large number of regions and a large number of types of regions. At the same time, news web pages are sufficiently similar in structure that a model can be learned from a relatively small number of pages. For segmenting these pages, we used our improved segmentation algorithm, as described in Section 3.3.

It is important to note that while the total number of *pages* is relatively small, the number of *regions* is very large—over 15000 in our dataset. The model predicts labels at a region level, and the empirical distributions learned by the model are defined in terms of individual regions and parent-child pairs. Thus the dataset is much larger than the number of pages suggests.

The distribution of labels in the dataset is highly skewed, as shown in Figure 4.6. This creates significant challenges in learning to classify regions. This is hardly surprising; a single page will only have one node in the `Page` class, but any given page may have dozens or hundreds of links; imbalance in classes, and the associated difficulties in classification, is an inherent aspect of the problem, rather than a matter specific to our dataset.

4.4.4 Results

The initial results of our improved experiments were unimpressive in terms of absolute accuracy. An overall accuracy of approximately 17% was achieved; in a 28-class classification problem, this is well above chance, but it is not a practical level of accuracy and is lower in absolute terms than the accuracy obtained in our earlier experiments (see Section 4.3). Some promising indications were, however, observed. Since the results were well above

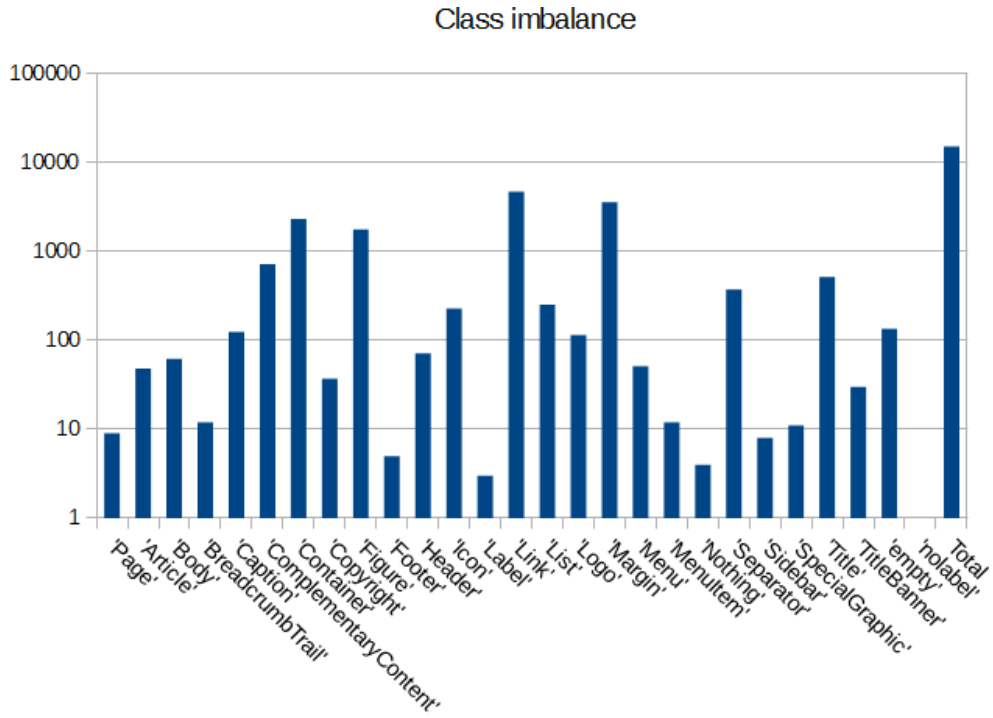


Figure 4.6: Class imbalance in the new web page dataset with manual labels. Note that the vertical axis uses a log scale; the degree of imbalance is very high.

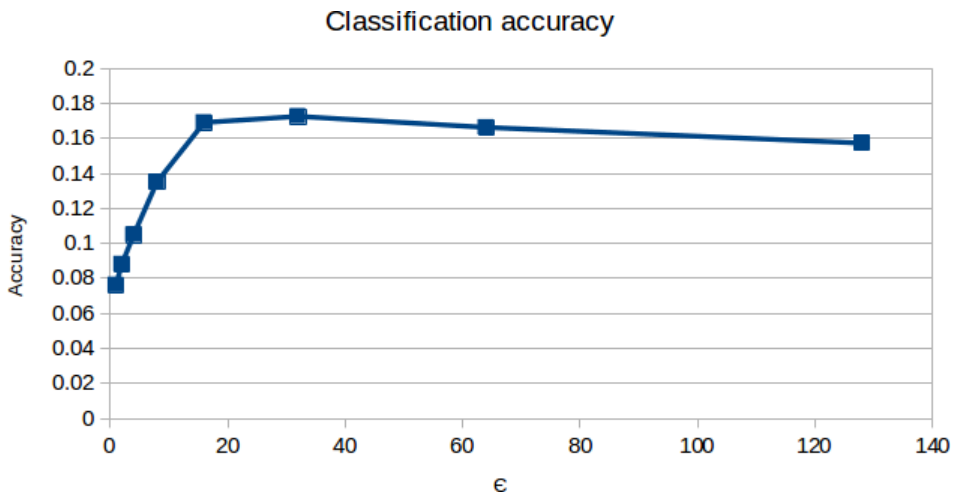


Figure 4.7: Graph of the accuracy of our classification algorithm as the parameter ϵ. Higher values of ϵ force a more uniform distribution over parent-child label pairs.

chance, this suggests that the learning algorithm is in fact learning even if the results are not reliable. As the value of ϵ (the parameter from Equation 4.2, Section 4.2.3 which raises the estimated probability of unobserved or rarely-observed parent-child combinations) was increased, raising the probability associated with rare parent-child label combinations, the accuracy initially rose, then fell gradually, as shown in Figure 4.7. This indicates the existence of an optimal value of ϵ, which is readily explained if we assume that these hierarchical relationships and features of regions both provide useful but not complete information in the classification process. Under this assumption, a very small ϵ will force the solution strongly toward common parent-child combinations, since rare combinations or those not seen in the training set have a very low probability in the learned distribution of parent-child combinations; the features of a region may not be able to overcome this low probability. Similarly, if the hierarchy is useful to the classification algorithm, a very large ϵ weakens the influence of the hierarchy by making the learned distribution more uniform; in the limiting case of a completely uniform distribution, the classification is based solely on the feature vectors describing individual regions. Thus, these results are not inconsistent with an algorithm that is learning *some* useful information about both the features of regions and their hierarchy.

Over-segmentation or under-segmentation of the page (with respect to the ontology being used) is a potential cause of poor performance in the classification algorithm, and can affect both the learned conditional distribution of feature vectors and of parent-child

combinations. It is simplest to examine the effects by dividing poor segmentations into several cases:

- When an under-segmented region is accurately segmented at its own level but not divided into smaller regions as it should have been, examples of feature combinations and relationships that should have been present in the training set will not be. If such under-segmentation is common in the segmentations available for the training set, classes that tend to occur at the lower levels of the hierarchy can be greatly under-represented in the training set.

- When an over-segmented region is accurately segmented at its own level but is divided into child regions when it should not have been, many features of its descendants can be distorted. The shapes and sizes of the child regions are not identical to the shape and size of the parent region, causing the proliferation of noisy features. Similarly, the proportions of different types of content can be differ from the proportions in correctly-segmented regions of the same class. Because the descendants of an over-segmented region will share its label, over-segmentation will also lead to a proliferation of instances of parent and child pairs sharing the same label, distorting the distribution of parent-child combinations.

- When a poorly-segmented region includes regions or parts of regions that should not have been grouped together, the ground truth label assigned to it does not accurately reflect all of its contents. As a result, the estimated features are distorted, by the different size, shape, and position of the composite region and by the different proportions of content types in the region. The distribution of parent-child combinations can also be distorted, since the poorly-segmented region may have child regions of a type that should not have been children of a region of its class.

Over-segmentation in particular can cause very large numbers of problematic child regions, since the node that should have been the leaf node can have a very large number of descendants.

The problems associated with learning reliable region features when the underlying segmentation is unreliable could, in principle, be addressed by selecting different sets of features. Features that describe the appearance of the entire region, not just its outline, would be more robust to segmentation errors than shape and size features if the appearance of a region is reasonably uniform. This approach has several important drawbacks:

- It cannot address the third type of poor segmentation listed above, since that type combines features from multiple classes that should not occur together.

- It only addresses the effects of unreliable segmentations on the conditional distribution of feature vectors; the distribution of parent-child combinations is still vulnerable to distortion.

- The features described in Section 4.2.2 were selected to reflect intuitively plausible features for distinguishing between different types of regions in a web page. It is reasonable to believe, for example, that a header is likely to be much wider than it is tall, while the opposite can be expected for a sidebar. Changing the features used could compromise performance by resulting in a set of features that is less informative even when regions are segmented accurately.

For these reasons, we turn instead to the structure of the segmentation tree to investigate possible solutions to the problem of low classification accuracy.

The segmentations produced for our classification experiments were produced using the same parameters as those used in concert with an assistive interface in the user study and corresponding offline experiments described in Chapter 5. Qualitatively, our segmentation divides regions quite aggressively with these parameters. Due to the nature of the assistive interface tested in Chapter 5, it was desirable to err on the side of over-segmentation, and aggressive segmentation was entirely appropriate. The requirements of the classification algorithm differ in that over-segmentation can be a cause of poor performance. To reduce the chances of over-segmentation, we used "pruned" segmentation trees, in which small regions were removed, to construct the HMT. Another means of reducing the aggressiveness of the segmentation algorithm would be to increase the minimum probability for introducing a new division from the default value of 0.5. From a probabilistic perspective, this is entirely appropriate; if the cost of a false negative is less than the cost of a false positive in a given scenario (such as producing a segmentation tree for later classification), the threshold should be correspondingly higher. For the purposes of this experiment, however, pruning has the advantage that the regions in the pruned segmentation tree are guaranteed to be a subset of the regions in the original segmentation tree; this avoids either time-consuming relabelling of regions or potentially error-prone association of labels between different segmentations. This is not the case for resegmenting with a higher probability threshold.

Figure 4.8 shows the effect of pruning the segmentation trees to remove nodes with a small area. The graph shows the accuracy of the classification process when the optimal value of ϵ (out of the values tested) is used. For the pruning process, we used eight region size thresholds: five, ten, twenty, thirty, forty, fifty, seventy five, and one hundred thousand pixels. Note that although one hundred thousand pixels seems to be a very large area, a 500×200-pixel region has this area and is a reasonable size for a mid-level

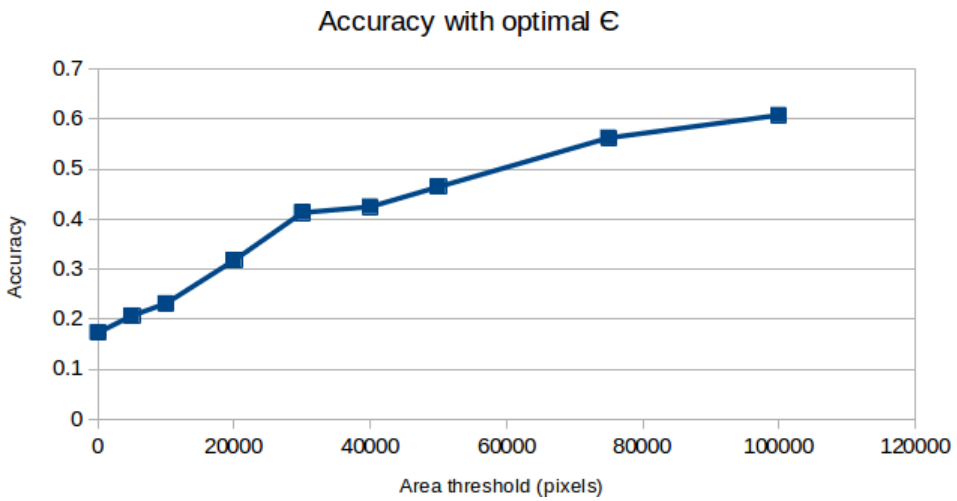


Figure 4.8: Graph of the accuracy of classification at the optimal value of ϵ (of those values that were sampled) as a function of area threshold value.

region in the segmentation tree. As the segmentation tree was pruned more aggressively, the classification performance improved. With the least aggressive pruning, removing only those regions with an area less than 5000 pixels, accuracy reached 21% with the optimal ϵ value, a modest improvement over the accuracy of classification algorithm on the unpruned segmentation tree. The accuracy with the optimal value of ϵ increased monotonically as pruning became more aggressive, as shown in Figure 4.8. For the highest area threshold, 1×10^5 pixels, the accuracy of the segmentation process reached approximately 61%, an impressive value for a multi-class classification problem with 28 highly imbalanced classes. As the accuracy had not yet ceased to increase at a threshold of 1×10^5 pixels, it is likely that higher accuracy could be achieved by further increasing the threshold, but at higher values the utility of the classification algorithm could be compromised as important mid-level features of the page are pruned away.

Figure 4.9 is analogous to Figure 4.7, and shows the effect of the ϵ parameter at each level of pruning. Regardless of how aggressively the segmentation tree is pruned, the accuracy is low for very low values of ϵ and rises (quickly, in most cases) until an optimal ϵ value; for values of ϵ above this optimum, the accuracy gradually decreases. Although the optimal value of ϵ varies somewhat with the degree of pruning, the qualitative consistency in the effects of this parameter is quite striking.

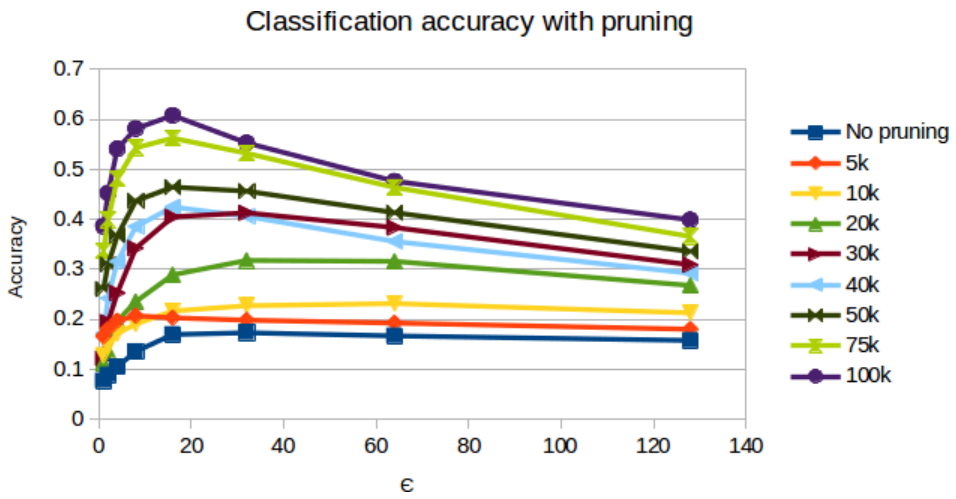


Figure 4.9: Graph of the accuracy of our classification algorithm as the parameter ϵ. Higher values of ϵ force a more uniform distribution over parent-child label pairs.

4.5 Applications

One of the most fundamental tasks users perform when browsing the web is locating information in the page that is relevant to their current needs and interests. When using a weather forecast page, for example, the desired information is probably the local forecast, either for the day or for the week. Other content on the page, such as features on weather events elsewhere in the country, is of secondary interest at best, and some content such as the copyright information in the footer is irrelevant to almost all users. Sighted users can perceive a large area of the page at once, organized in two dimensions, to quickly locate large, high-level content areas to skim for the relevant information. Although users with visual impairments severe enough that they must use a screen reader to access the page do not have this immediate access to the two-dimensional layout of the page, they too are generally not interested in all content on the page, but will want to skim or skip various parts of the page [109]. Screen reader users have a much more difficult time making these macro-level assessments [136].

To mitigate this challenge, screen readers typically include features enabling the user to navigate (i) from heading-to-heading or paragraph-to-paragraph; (ii) through each item in a list with advance notification of the length; (iii) through data tables, cell-by-cell, with headings; (iv) alphabetically through all the links on a page. These features enable the user to hear an outline of the page's main ideas and then focus on the most relevant parts. If one

examines sample transcripts from screen readers[4], there are examples of rather complicated navigation methods presented to users.

Aside from the complexity of the DOM tree-based navigation interfaces, the efficacy of this approach depends heavily on the quality of the underlying code. Even though elements might look appropriately formatted to a sighted user, if they are not properly defined in the HTML, they will not be available to the screen reader. Moreover, if a page uses the same style for a mix of elements on the page (for example, if a heading tag is used both for the headings in the main content and for navigation elements), the screen reader will not be able to differentiate. HTML5 sections and ARIA landmarks can address this problem by clearly demarcating different sections of the web page (*e.g.*, headers, navigation, main content, *etc.*) but unfortunately these markers are often absent, and even when present they may not be correctly or consistently used. Using computer vision to automatically classify regions in a web page according to their role can help to address the poor support for screen readers by making high-level semantic labels available[5] without relying on developer support and independently from the implementation of the page.

For some sighted users, the ability to see a large amount of information on the page simultaneously can be a drawback. Older users [14] and users with cognitive or attention deficits [85] may find the complex, busy layouts of many modern web pages distracting. This can interfere with effective use of web pages. Segmentation alone can support some types of assistive interface to reduce distractions. As shown in Chapters 3 and 5, the segmentation tree can be used to highlight a focus region relative to the surrounding areas of the page in order to make the focus region more salient than potentially distracting regions. When combined with classification, more sophisticated approaches are possible. Prediction of the most relevant regions based upon the labels could allow irrelevant, distracting regions to be automatically suppressed. This would be advantageous, since the user would have less need to manually indicate a focus region, and the entire page would be simplified.

[4] *e.g.* NVDA, found at `nvaccess.org` [97]

[5] Available directly to the screen reader and potentially indirectly to the user through their use as navigation landmarks

Chapter 5

Assistive Interface User Study

This chapter describes an experiment aimed at validation of the practical utility of the page segmentation method described in Chapter 3 as a back-end parsing system to support a front-end assistive interface. This technique was developed in order to leverage AI, and in particular computer vision algorithms, as part of an effort to provide assistive technology for online users who may have visual or cognitive impairments. It is of interest because the use of images of web pages rather than the DOM tree or page source code provides complete implementation independence, and does not rely on inconsistently-used accessibility frameworks.

The tests of our segmentation algorithms presented in Chapter 3 focused on two primary types of experiment. Quantitative evaluation of the approach, in comparison with a competitor tied to DOM-tree evidence [22] and with a DOM tree-based control algorithm, demonstrated that the vision-based segmentations are similar to DOM tree-based segmentations at a large scale but offer important differences at a small scale. In these experiments, an earth-mover's distance metric (see, *e.g.*, [114]) was used to compare the leaf-level segmentations. Qualitative evaluation of segmentation quality established that the vision-based segmentations captured important features of the pages on which they were tested, and that the overall segmentation quality appeared good.

In this chapter, we focus on the challenge of validating the performance of our visual segmentation method in a system complete with an assistive interface through a user study with participants from a population likely to require assistive technology and through offline experiments with the capabilities of the interface. This study was approved by the University of Waterloo Office of Research Ethics. We discuss in detail the design of this user study: the tasks presented to users, the data set used, the metrics tracked, the participant

responses elicited, and the interface design. Lessons learned reveal directions for further experimental design. This research therefore provides insights for other researchers on how to bridge the gap between the many plausible computer vision algorithms and their *practical* value to actual users of the algorithms. It also discusses the particular challenges which must be addressed when these users have assistive needs. We note that user study results are lacking for the broadly similar vision-based page segmentation algorithms described by Barkol *et al.* [12] and Cao *et al.* [24]. The discussion we include in Section 5.3 sheds light on the difficulties that arise when designing interfaces for a study of this type, and will also reveal a role that can be played by offline testing of the proposed designs.

5.1 Experiment Design

Testing the effect of the back-end system used for parsing web pages on the performance of assistive technology requires particularly careful experimental design, due to the presence of the front-end interface between the back-end visual parsing system and the user. We use two related experiments to address the problem: one in which users perform a simple visual search task, and an offline test of the capabilities of the system. Visual search (*i.e.*, the user looking for a target in the page) is a sub-task common to many web browsing tasks (and therefore should prove informative about overall benefits), but it is sufficiently concrete and quantifiable to allow good experimental control and reduce the complexity of the interface required for the task.

In the user study, the participants complete a series of tasks in which they are presented with a news web page and a description of a target article to find using either a "vanilla" non-assistive interface or an assistive interface. The assistive interface allows users to magnify and highlight a user-specified target region (in order to accomplish zooming and decluttering) and can be customized by the participant in a separate configuration step. Magnification and highlighting are implemented through image editing operations on an image of the page. Figure 5.2 shows an example of the interface in use. The performance of participants is measured in terms of speed and accuracy in finding the target article, and data about participants' subjective evaluation of the assistive interface is collected using brief questionnaires at various stages during the experiment.

For the purpose of this experiment, the assistive interface presented to users needed to be tied to the output of our segmentation system. In so doing, we would be able to determine whether our framework properly supports assistive features. Since our segmentation algorithm produces a segmentation tree identifying the regions of a web page, we designed the study in order to try to represent what users interpreted as a region of interest on

the page. In brief, we chose to use an interface that allows the page to be displayed with selected regions magnified and highlighted, using the segmentation tree to ensure that the regions correspond to semantically significant, visually coherent regions of the page. Magnification and highlighting were chosen because they had been noted to be appropriate for use with a segmentation alone [35].

In the offline tests, we used simulated "noisy" user input to evaluate the accuracy of the estimated region of interest. Offline testing uses simulated user input as a stand-in for real participants; it allows the theoretical capabilities of the system to be thoroughly explored, and is unaffected by any user experience issues that may occur in a simple test GUI.

5.1.1 Interface Design for Visual Search Study

As indicated earlier, this experiment is concerned with identifying a user's true region of interest, based on a rough selection, as the user searches the page for a target article. Given a segmentation tree expressed as a graph (S, E) (where S is the set of regions in the segmentation and $E \subset S \times S$ is the set of directed edges from parent to child regions) and a user-selected approximate region of interest R, the interface estimates an intended region of interest R' consistent with the segmentation tree by optimizing over some measure of accuracy. This measure of accuracy should be reduced by both false positives (pixels outside of the user's selection that are included in the estimated region) and false negatives (pixels inside the selected region that are not included in the estimated region).

In our experiment, we defined the estimated region to be $R' = \bigcup_{s \in S'} s$ for some $S' \subseteq S$; the set S' is chosen such that $S' = \text{argmin}_{S'} \frac{|R' \cap R|}{|R' \cup R|}$, subject to the condition that $\forall s_i, s_j \in S'$, $\text{depth}(s_i) = \text{depth}(s_j)$. In other words, R' is the best approximation (in terms of the accuracy function described) of the user-selected region of interest produced from a set of sibling nodes in the segmentation tree. Alternative formulations could include using an F-score measure of accuracy or relaxing the requirement that all nodes used to build the selected region be siblings; these may be worth evaluating in future experiments.

The segmentation tree allows the system to display a visually coherent region of interest even when the user's selection did not correspond well to his or her intended focus region. Such an imprecise selection could occur for a number of reasons: because the user has difficulty with using the input device (*e.g.* a user who has difficulty using a mouse due to hand tremor), because the user requires an assistive input device with low resolution, or simply because the user is in a hurry. Evidence from a study conducted during a computer class for older adults by Dickinson *et al.* [40] shows that using the mouse (especially

for more complex tasks like clicking and dragging) can be challenging for older users, particularly those with specific challenges in fine motor control. Although these difficulties can often be overcome with practice, they are a source of frustration for novice users [40]. Inferring the intended region, rather than simply using the selection made by the user unquestioningly, may help to address these problems.

5.1.2 Participants

Older adults—generally considered to be 60 and older—can potentially benefit greatly from the web and the internet, but the needs of these users are often not well served by current designs [92]. Existing research indicates that older users browse differently from younger users [14, 47, 123]. Older adults can have complex assistive needs as well as different methods of processing a web page from younger users, and represent a growing segment of the population [62]. For users with specific assistive needs, many frameworks and standards have been developed (*e.g.* WAI-ARIA [128]), and aspects of these frameworks are applicable to designing for older users [9]; unfortunately, these are not reliably used even on popular web sites [33]. We show that using computer vision to interpret web page structure is a promising approach, and the experiments presented here are intended to test a specific case of this more general hypothesis.

For our user study, we recruited a total of 16 participants, all older adults. All of the participants lived in the community; their degree of independence helped with logistical and ethical issues in conducting the experiment, but may also have resulted in users with less need for an assistive interface. The first three participants used an earlier version of the assistive interface in a pilot study; the results of the pilot study suggested user experience problems with the front-end interface, which the redesigned graphical interface used in later experiments was intended to address.

5.1.3 User Study Protocol

In our experiment, participants were asked to locate a specific article in a cluttered news page as quickly as possible, based on a description of the article slightly paraphrased from the headline and description occurring on the original page. News web pages were chosen because they are a popular class of web pages, with content that is updated frequently (thus facilitating the generation of a large dataset of unique, but related, pages). The example pages used were collected well before the users participated in the experiment, to prevent users from remembering where to find a specific article; similarly, no page was

shown more than once during the course of the experiment. Web pages were chosen from BBC News, CBC News, and New York Times web sites, and the dataset included examples of the home page and of high-level sections such as "World" and "Business". Although the task shown here uses news web pages, similar visual search tasks are found when finding a post about a specific topic in a social media page or finding a product of interest on a shopping web site.

Our participants were recruited through word-of-mouth and through a participant pool of older adults affiliated with the University of Waterloo. Each participant saw a total of 30 images over the course of a one-hour session; a mechanism was included in the test to end the experiment with fewer pages if the total time exceeds a preset limit, but this was never necessary. On occasion, an example would be skipped due to participant frustration. The images were divided into two equal sets, and for each user one set of was to be viewed with the option to use assistive features and one to be viewed without. Order of presentation of the assistive and non-assistive interfaces was counterbalanced (*i.e.* which one to be shown first to each user) to mitigate learning and fatigue effects.

Articles were selected such that each had at least two of the following characteristics: **1.** A headline or title, **2.** A graphic (or placeholder), and **3.** A description of the article. Additionally, articles were chosen such that they appeared only once in a given page to ensure that the target was unique. For each article, the corresponding ground truth bounding box was manually labelled[1]. The ground truth bounding box was not selected to correspond to a region or group of regions in the segmentation tree of the page produced by our algorithm, since if a region was segmented poorly, the resulting bounding box could falsely accept or falsely reject some responses from the user, and if only very accurately segmented targets were used, the results would be biased. In selecting targets and drawing target bounding boxes to establish ground truth, we did not consult the segmentations at all, in order to avoid biasing the set of target regions. Figure 5.1 shows examples of the target articles used and their descriptions. We selected articles from a variety of regions of the page (including different columns and vertical positions) and in a variety of sizes to ensure that the sample was reasonably representative. Similarly, the dataset contains pages from several different web sites, with different lengths and numbers of columns. To ensure similarity between the sets of pages used in each condition for each user, the pages were divided such that distributions of article positions along the length of the page were similar. The height of target regions ranged from 111 to 411 pixels, with a mean of 228.7, and the width ranged from 196 to 666 pixels with a mean of 342.8. All participants used the same computer and monitor in the test; the 22-inch monitor had a 1680×1050-pixel

[1]We knew where on the page the target article was located; we identified this in order to determine whether users successfully located the target or not.

(a) "The inventor of the Heimlich maneuver saves a woman from choking"

(b) "Removal of statues of Lenin in Ukraine"

Figure 5.1: Example targets and the corresponding descriptions provided to the users as prompts, from our experiment. Both examples are taken from BBC News pages.

resolution, and the viewable area of the page was 1024×768 pixels.

We selected older adults for our target population based on existing research that suggests that an interface of the type tested could be of use to this population [85]. Aside from the obvious applications of magnification (equivalent to large-print books), there is evidence to support the utility of some mechanism for decluttering. Eye-tracking research has shown that older adults attend to areas on the page longer than younger users do [123], and that they are more sensitive to clutter and distracting elements [14]; as a result, distractions from irrelevant elements of the page may waste more time for this group. Highlighting a focus region relative to the rest of the page may help users to avoid distractions.

For testing purposes, we implemented an instrumented version of the interface described above in a dedicated system. This version of the interface, depicted in Figure 5.2, includes experiment-specific controls and logs all interactions for later analysis as well as collecting speed and accuracy data. The web pages used in the testing procedure were rendered as images, because interactivity was neither required nor beneficial and because image processing operations could be used to easily implement the interface operations. In the interest of ensuring that all controls and instructions are legible to users who may have

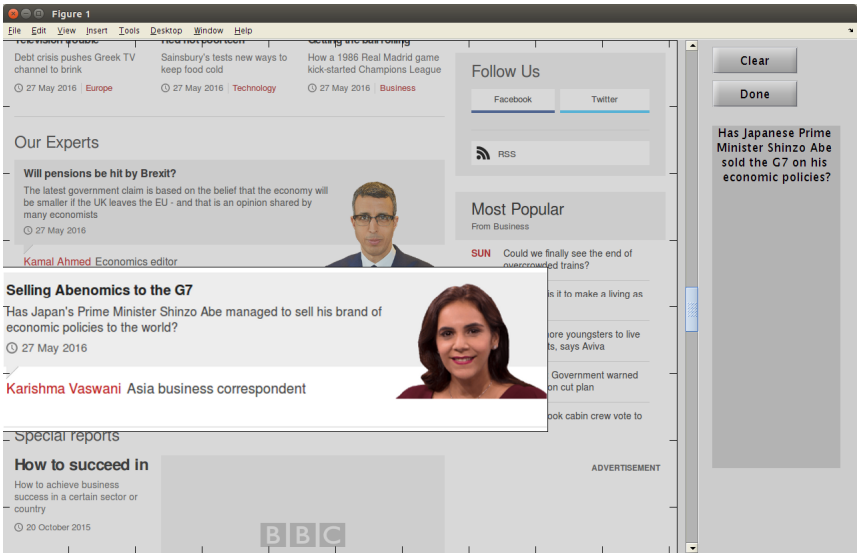


Figure 5.2: The assistive interface being tested, shown here with the correct article selected as the focus region. This interface shows the default levels of magnification and highlighting.

some degree of visual impairment, all interfaces were designed to use large font sizes. This interface consists of two parts: first, a configuration utility in which users set the degree of magnification and highlighting to be used for the rest of the experiment[2]; and second, a simpler interface with fixed magnification and highlighting in which the user actually completes the tasks. For comparison purposes, we also implemented a "vanilla" interface with no assistive features in the same framework. This was chosen as the point of comparison because our objective is to determine whether or not a purely vision-based framework can support a useful assistive interface. At the present stage of development, it is more informative to compare against a control with no assistive features than against a state-of-the-art DOM tree-based method, since those methods have had a considerable period of iterative refinement, while vision-based methods have not.

In addition to logging activity and measuring visual search performance, we collected data about participants' subjective impressions of the interface and their familiarity with the web. Before beginning the visual search tasks, the users were presented with a brief questionnaire about preferences and background, including the following questions:

[2]This also provided an opportunity to demonstrate the capabilities of the interface to the users and for users to practice using the interface.

1. I use the Internet regularly (rated on a five-point Likert scale from "Strongly disagree" to "Strongly agree")

2. I find it frustrating to use the Internet (rated on a five-point Likert scale from "Strongly disagree" to "Strongly agree")

3. Select the text size you would prefer to read (options show a range of font sizes)

After each interface, the users are presented with a standard NASA TLX workload evaluation questionnaire[3] [59] to allow comparison of perceived workload between the assistive and vanilla interfaces. Finally, users are presented with a brief questionnaire to elicit information about preferences between the two interfaces (vanilla and assistive) tested. This consists of the following questions:

1. Which interface was easiest to use?

2. Which interface was most frustrating to use?

3. Which interface would you prefer to use?

Collecting separate information about efficiency (time to find the target), effectiveness (error rate), and subjective preferences is important; since the study is intended to detect any positive effect produced by an interface using the vision-based segmentation system, an improvement in any aspect of usability would be considered a positive result. In a meta-analysis conducted by Hornbæk and Law [69], these aspects were found to be positively correlated in most studies, but only to a low or moderate degree. This indicates that separate measurements are valuable for our purposes.

5.2 Results

In this section we present the results of a series of experiments based on the design described above. In Section 5.2.4, we also discuss offline experiments performed using our interface without user involvement.

A small pilot study with three participants helped to inform the design of the interface used in the study proper; the feedback from these participants helped to refine the interface design and eliminate inconvenient or unintuitive aspects of the control scheme.

[3] Shown in Appendix B.

The modified interface was tested with eight participants; based on the results, a new set of tasks was created and a further five participants recruited for a second phase. Offline testing was also performed to more fully explore the capabilities of the interface. Of these experiments, the offline tests proved to be the most informative about the combination of the segmentation algorithm and the assistive interface.

5.2.1 Interface Design Iterations

The initial design for the interface used a single click to control the position of the area of interest and a slider to determine its size, rather than the click-and-drag selection method used in the final version of the interface. Although researchers were able to use the system efficiently, and participants were given a tutorial on using the interface, users tended not to use the size control, even though one commented informally that the region was too small. We also observed a tendency to ignore the assistive features; including the necessary click on the target region, 26 out of a total of 41 completed examples involved two or fewer clicks. From this, we concluded that the interface was not sufficiently intuitive or convenient to be useful for the purposes of this study. The interface was then modified to use the region selection method described in Section 5.1.1.

5.2.2 Results from Vanilla Interface

Since the vanilla interface functioned identically in the pilot study and in the final design, it is possible to combine the data from both stages to verify that the visual search time for each instance is consistent with reasonable assumptions about the methods that participants would use to find the target. Figure 5.3 shows the relationship between the vertical positions of targets in the page and the time taken for participants to find the target region. Beyond the area of the page visible on the first screen, the data shows an upward trend in the minimum search time to find an article. A more formal analysis using linear regression gives a slope of 0.037 s/pixel, or 27 s per screen-height, and an intercept of 6.3 s; the slope of the relationship between position and time is statistically significant for $\alpha = 0.001$. This supports the assumption that participants would search the page from top to bottom. The maximum time shows no clear trend, which can be attributed to the fact that articles that take a long time to locate were often missed when scrolling down the first time.

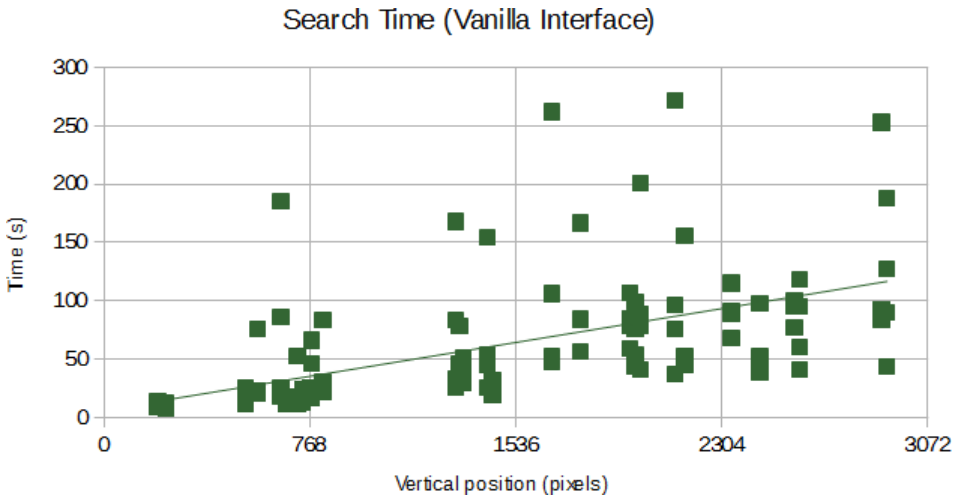


Figure 5.3: Time to locate target regions as a function of their vertical position (measured from the top of the page to the top of the target) in the first phase of the user study. Note that the minimum time to find articles shows an upward trend as more scrolling is required, as expected. The horizontal axis is divided into sections the size of one window.

5.2.3 Main User Study Results

Our user study provided results which, although not conclusive with respect to the efficacy of the interface itself, are of considerable interest. In terms of search time, there was no significant difference between performance with the assistive interface and performance without it. Because the search times for different pages are not directly comparable, we compare the average times for each page with and without the assistive interface; over the set of 30 web pages, the average time improved with the assistive interface for exactly fifteen pages; on average, the search time was 1.93 seconds longer with the assistive interface than without, with a large standard deviation of 30.3 seconds. Neither represents a statistically significant change (using a t-test, $t(29) = 0.349$, $p = 0.73$). This is not surprising, as participants made relatively little use of the assistive features: in 78% of examples using the assistive interface the only interaction with the page was a single click on the target.

The results from the surveys also do not show an improvement with the assistive features. Table 5.1 shows the results for subjective preferences between the two interfaces. The responses for each dimension of the TLX survey (Table 5.2) are more mixed. There is a general tendency to prefer the vanilla interface, but given that the assistive interface functions identically unless assistive features are specifically invoked this could be a matter of unfamiliarity.

	Vanilla	Assistive	Neither
Easiest	6	1	1
Most frustrating	1	4	3
Preferred	6	0	2

Table 5.1: Summary of subjective preferences from first-phase tests.

	Vanilla	Assistive	Neither
Mental demand	4	2	2
Physical demand	4	2	2
Temporal demand	2	4	2
Performance	2	4	2
Effort	5	2	1
Frustration	6	2	0

Table 5.2: Table showing the number of users for which the assistive or vanilla interfaces were better according to each dimension of the TLX questionnaire.

Because the results from the first round of participants were inconclusive, we also produced a second, more difficult set of tasks. For these tasks, the main considerations were that neither headlines nor images should allow the user to guess the target without reading smaller print. The idea was to make the task of locating the specified article more challenging, with the thought that some users might then have greater motivation to use the assistive features. As the tasks were made more difficult, the total number of examples was reduced from 30 to 20. We also added minor convenience features, such as a button to clear the magnified and highlighted region. The results here were similarly inconclusive; of 45 completed tasks using the assistive interface, 53.3% involved only a single click and 73.3% involved no more than two clicks; given the close proximity of many clicks, not all necessarily represent a deliberate click-and-drag action. Due to low engagement with the assistive features, this phase was cut short in order to focus on offline tests.

We believe that there are several key problems in obtaining conclusive data. If a potential participant has assistive needs that make a computer difficult to use with existing assistive technology, he or she may be disinclined to participate in any computer-based study at all. If he or she does decide to participate, inexperience with using a computer may cause the participant to struggle with *any* interface in the course of a sixty-to-ninety-minute experiment. One user made informal comments to this effect. On the other hand, users who can use a computer effectively without assistive technology are likely to be

very open to the idea of participating in a computer-based study, but would not benefit significantly from the assistive interface; in our recruitment pool the records of participants' histories were not sufficiently detailed to determine which potential participants would need an assistive interface. The recruitment process may introduce a participation bias even if the population contacted is representative of older adults in general.

Information gathered through the background survey (described in Section 5.1.3) about computer use experience supports the hypothesis that the recruiting process was biased toward those who did not need the assistive interface: of the 13 participants after the pilot study, only 1 disagreed with the statement "I use the Internet frequently", and only 4 agreed with the statement "I find it frustrating to use the Internet".

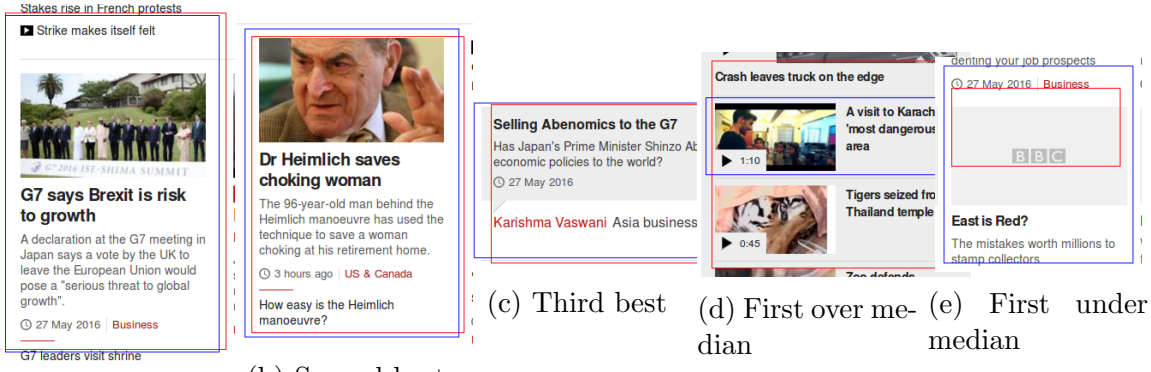
If users had employed our assistive features more extensively and had responded that they found this to be a less frustrating solution, maybe even with better completion times for locating articles, we could have concluded first that our framework is supporting assistive features well (zooming and decluttering), and second that the regions generated by our segmentation algorithm were generally helpful in support of such an interface and therefore the methods employed here were well designed.

Because this did not happen, we opted to proceed to a phase of offline testing. For this experiment, there are different metrics involved in order to reach conclusions about the value of our framework.

5.2.4 Segmentation Quality

To fully explore the capabilities of the combined segmentation and classification system, we made use of simulated input in a series of offline tests. In essence, we created simple "virtual users" to simulate input to the interface in order to exercise its capabilities. This allows a more comprehensive test of the theoretical performance of an assistive interface, while providing independence from user experience concerns with the interface itself, and avoiding the need to obtain enough participants who require an assistive interface to thoroughly test the performance of our segmentation method and its interpretation by the front-end interface.

Our approach to offline performance evaluation is, necessarily, different from the approach taken with human participants. It is, to say the least, infeasible to create a virtual user capable of accurately simulating the behaviour of a human participant searching for a target article in a news page. Fortunately, this is not necessary. In order to evaluate the ability of our page segmentation algorithm to support the type of assistive interface in question, we need only evaluate the ability of the segmentation to support inference



(a) Best match (b) Second best (c) Third best (d) First over median (e) First under median

Figure 5.4: Examples of target regions and corresponding automated estimates of intended regions of interest. The left three show the best matches; the right two examples show cases with approximately the median match quality (as there were an even number of examples, the median quality does not correspond to a single image). The target region is shown in blue and the estimated region of interest is shown in red.

of the true region of interest from the user's rough selection; all of the influence of the segmentation upon the performance of the front-end interface is mediated by this step. To assess the performance of our method in this sense, we use ground truth regions of interest corresponding to articles in the example pages; these are plausible regions of interest for a real user. The virtual user's input is produced by adding noise to the bounding box of the ground truth region of interest to simulate a rough selection.

Support for the quality of the segmentation produced by our back-end system can be seen from an analysis of the accuracy with which the ground truth region of interest can be isolated in the segmentation tree. To test the level of performance possible for our segmentation algorithm (as shown in Section 3.3) on the dataset shown to users, we applied the region of interest estimation procedure to a hypothetical selection corresponding exactly to the target region (*i.e.*, the virtual user selects the region of interest with perfect accuracy). The resulting estimated regions of interest show good performance for the region of interest estimation procedure, especially for the most accurate cases. Figure 5.4 shows example comparisons between the target region (blue) and the estimated region (red). Note that the best results show regions segmented almost exactly, and the median results show major parts of the target region accurately segmented. These results strongly suggest that our algorithm does produce high-quality visual segmentations, because the true regions of interest correspond well to their closest approximation in these segmentations.

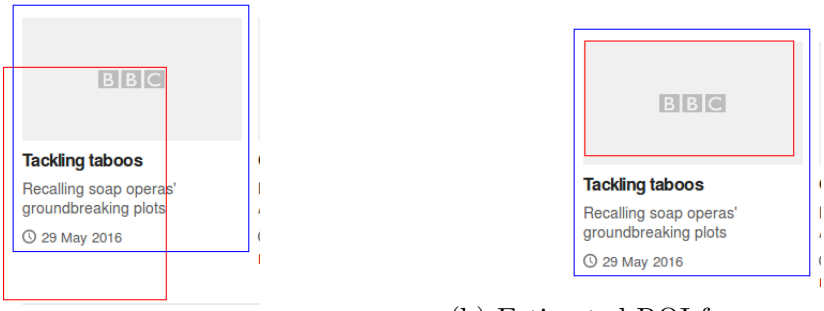
Our virtual user can simulate inaccurate selections just as easily as accurate selections. Since the advantage to combining click-and-drag region selection with the segmentation tree lies in the ability to overcome poor selection accuracy, this is an important aspect of evaluating the performance of the back-end system. To simulate a noisy or inaccurate selection, we perturb the boundaries of the true region of interest by adding normally-distributed noise to the positions of the boundaries. The standard deviation σ of this distribution is the only parameter in the noise (for the sake of simplicity, we use identical distributions for all four sides). Note that this is a simple model, but it allows the generation of a wide range of potential inputs to thoroughly exercise the capabilities of the interface. By adjusting σ, we can control the accuracy of the simulated input.

If our system always produced the true region of interest, regardless of how noisy the input was, this would mean that our system would be performing perfectly, inferring exactly what the user's region of interest was, even from only a vague hint of what it was. Since it is not realistic to expect perfect accuracy regardless of the level of noise, we wish to determine how close we can get to the true region of interest, and how gracefully the performance degrades as noise increases. In particular, if the quality of our estimated ROI degrades more gracefully than the quality of the noisy segmentation, then our framework will ameliorate the effects of poor selection accuracy on the performance of the interface.

We tested the quality of estimated regions of interest produced using normally-distributed random perturbations of the target region boundaries as hypothetical selections. The quality of these estimated regions of interest was measured using the intersection-over-union measurement of agreement with respect to the ground truth and estimated regions of interest. For each of the 120 targets, 30 perturbed selections were tested for each σ value for the perturbations of the boundaries of the simulated selection. These simulated selections were evaluated for agreement with the ground truth region of interest, just as the estimated regions of interest were.

Figure 5.6 shows the quality of the estimated regions of interest and the simulated noisy selections (on the vertical axis) as a function of the standard deviation σ of boundary position error. This graph clearly shows that over the range of σ values considered, the quality of the estimated region of interest is nearly flat; this indicates steady or gracefully degrading performance. As the noise in selection position increases, the quality of the noisy selection drops much more rapidly and quickly becomes comparable to the quality of the estimated region of interest at most levels of accuracy. The best results from the interface quickly exceed the best noisy selections (see, for example, the 90^{th} percentile curves, which intersect at $\sigma \approx 25$).

An additional consideration is whether or not these simple measurements of accuracy



(a) Noisy selection

(b) Estimated ROI from segmentation

Figure 5.5: Comparison of two estimated regions of interest: on the left, a noisy selection of the type used for offline evaluation, and on the right an estimated ROI based on the segmentation. The two examples have approximately the same accuracy score (0.42 and 0.41). Note that the segmentation accurately isolates part of the region of interest, while the noisy selection is inaccurate for all parts of the region.

accurately convey the semantic differences in quality; Figure 5.5 contrasts the types of error encountered with the estimated ROI and the noisy selection in cases with approximately equal quality. The estimated region of interest is visually coherent but at a different level of organization than the true region, while the noisy region does not accurately reflect any aspect of the region of interest.

Our procedure for estimating the true region of interest based upon a segmentation tree and a noisy selection requires a high-quality segmentation tree in order to accurately reconstruct the true region of interest. The quality of the estimated region of interest cannot exceed the quality of the closest match to the true region of interest found in the segmentation tree. Furthermore, the closest match in the segmentation tree to the noisy selection must be similar to the true region of interest. This prevents a trivial segmentation in which the page is divided into single-pixel regions from producing results better than the noisy selection. Our results show that the estimated region of interest improves upon the quality of the noisy selection; as a result, we can conclude that the segmentation tree corresponds sufficiently well to the structure of the page to be useful for our example interface, especially for the most accurately segmented regions. At lower quality levels (for example, the $25th^{\text{th}}$ percentile and below), the performance of the region of interest estimation method is noticeably lower in absolute terms. Improving the worst-case performance of the segmentation algorithm is, therefore, an appealing direction of research, while the best-case performance appears to be very good in its current state. One approach to improving worst-case accuracy may be to use a more sophisticated structural

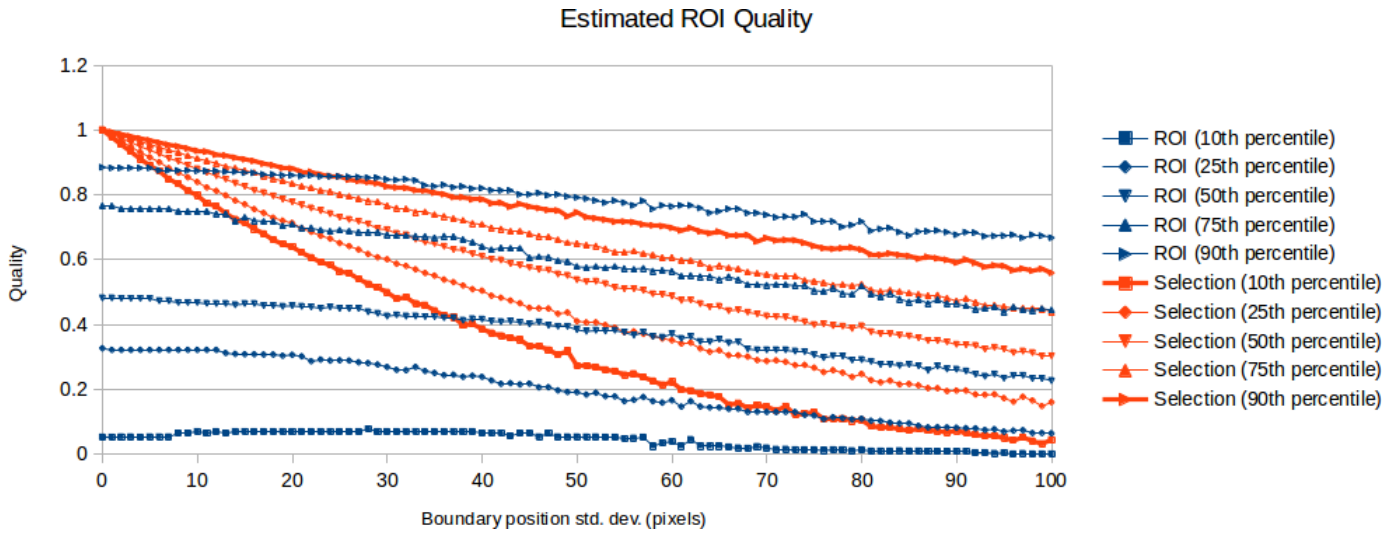


Figure 5.6: Quality of fit of estimated ROI based upon our segmentation trees (blue) and original noisy Selections (orange) as a function of the standard deviation of selection position error. Shown are the 90^{th}, 75^{th}, 50^{th}, 25^{th}, and 10^{th} percentiles of quality. Note that the quality of the estimated regions of interest based on the segmentation trees remains nearly flat as the quality of the selection decreases.

prior; although effective, the X-Y tree segmentation prior only reflects a limited view of "web page-like" structures, and improvements here could address errors caused by a mismatch in granularity as shown in Figure 5.5.

5.3 Discussion

One of the key challenges in our experiment is recruiting suitable participants. Because of the requirements for describing the experiment in recruiting materials, potential participants are aware that the experiment is computer-based. The users who are most likely to benefit from an assistive interface of the type being tested may, however, experience frustration with using computers and therefore be less likely to respond or consent to a computer-based study. This could result in the introduction of participation bias toward a negative result even if the potential participants contacted represent an unbiased sample from the population of older adults. Strengthening recruitment criteria may reduce the number of participants who do not require any form of assistive interface, but could exclude participants who would benefit for reasons of preference rather than of need. It may be useful to recruit even older participants (*e.g.* 80+); there are indications that the types of challenges for which assistive interfaces may be useful could be more common in an older group (see, for example, the discussion of the effects of aging in [62]). Although we recruited participants through a combination of word-of-mouth, email contact, and phone contact, most had email addresses and were familiar with computer use.

Although this test is focused on assistive interfaces for older adults, we hope that similar methods might be of use to other users. Even users with no assistive needs may derive benefits from similar technologies through better methods for altering the presentation of pages to be more convenient to use. This is an area that merits further study. Also of interest is the possibility of using vision-based parsing (especially in a complete segmentation-classification pipeline) to improve within-page navigation models for screen-readers such as NVDA [97]. This is likely to require more extensive experiments than the proof-of-concept study presented here, since it would require combining the visual parsing system with an appropriate navigation model and evaluating the complete system in real-world conditions.

There has been extensive research into the design of assistive interfaces and the various transformations of the screen contents that can be used to provide an appropriate interface to users with assistive needs. In the more restricted domain of appropriate modifications for older users, we have cited several papers in reference to the design of our interface. Also of note is a paper by Hwang *et al.* [71], which presents a comparison between the performance

of older adults and younger users with using interfaces that dynamically magnify targets under different conditions (*e.g.* targets that are magnified when the cursor approaches the target and targets that are magnified only when the cursor enters the target region). Moving forward, it may be of interest to test interfaces using such variations on the basic concept of a magnifying interface in our experimental setting and compare the results.

Chapter 6

Segmentation Study

The examination of web pages as visual images, in order to support tasks such as decluttering or zooming of information on these pages, has attracted increasing attention in recent years. These tasks are of particular value for users with cognitive or visual assistive needs. One of the key sub-problems in this area is web page segmentation: the problem of dividing a page into coherent, semantically meaningful regions.

Interpreting web pages as images enables implementation independence, in contrast with approaches to web page segmentation that rely on the DOM tree[1]. Chapter 3 described our promising approach to this segmentation problem, which adopts a Bayesian solution to producing a segmentation tree, based on successive detection of locally significant edges, semantically significant lines, and ultimately visually coherent regions.

The initial validation for the vision-based page segmentation methods in Chapter 3 focused on qualitative evaluation for the overall quality of segmentations; quantitative measures were restricted to checking for agreement between algorithms (see Section 3.2) and tests of the accuracy with respect to small numbers of target regions in a practical context (see Chapter 5). These evaluation methods were useful and provided important information used in developing the later version of the segmentation algorithm. Ultimately, however, it was decided that a ground truth segmentation dataset would be of value, since it could be used to provide more definitive, measurable evaluation of overall segmentation quality. This chapter describes how these ground truth segmentations were acquired, in a user study where participants were asked to manually draw and edit web page segmentation trees, and the properties of the ground truth dataset obtained. This study was approved by the University of Waterloo Office of Research Ethics.

[1] An intermediate representation of the page produced by the browser in the rendering process.

Our exploration of ground truth segmentation for this particular domain leads to more general insights of value for the computer vision community. In particular, we explore the best methods for eliciting ground truth and what the characteristics of those ground truth segmentations are. Our approach contrasts with the methods used to build the well-known Berkeley segmentation dataset (BSDS) [90], first published in 2001 and widely used even today to test segmentation algorithms designed for natural images. In particular, we aim to have a larger base of users per image though we also make use of a smaller set of images; this is done in order to gain greater insights into agreements (and disagreements) between users for a given image, which allows us to estimate the ceiling on the performance of a one-size-fits-all solution, as well as to determine the most efficient way to gather a large dataset of segmentations (*e.g.*, how many users should segment each page to get a good sample of segmentations while avoiding unnecessary duplicated effort).

To accomplish our objectives for this study, we designed a segmentation editing tool which allows users to both specify their perceived segmentations from scratch and edit an existing segmentation; this tool allows us to investigate whether users differ in their interpretations when grounded in the same specific starting point, and to what degree the starting point affects the perceived ground truth segmentation.

Users annotate web pages to indicate where they perceive bounding boxes or regions; from these, we are able to build a representation of that user's view of the web page which specifies various structural properties. Nested bounding boxes implicitly define a hierarchy where the containing region is the parent of the contained region. This interpretation allows us to draw conclusions about the structural properties of the page, as viewed by this user, and to compare these interpretations across users as well. In so doing, we are able to have the basis for imposing a prior probability distribution on segmentations, providing important directions for the design of segmentation algorithms in this domain. We note that for the domain of segmenting web pages, it is possible to start with some constraints on regions. Specifically, it is reasonable to assume that regions are axis-parallel rectangles.

In the sections that follow, we provide some important background on both segmentation research and on efforts to interpret web pages in particular. We then describe in detail the user study that we performed, the editing tool we designed, and the design decisions we made about what tasks to give to users and how to analyze the results. This includes some explicit examinations of similarities and differences between users. Included here are valuable methods for overlaying alternative interpretations of segmentations, in order to facilitate the comparisons. From here we reflect on the value of this effort to derive ground truth and the potential applications of these results for the future.

6.1 Study Design

This study is based on the collection of segmentations produced manually by a significant number of participants (a total of 29 participants[2]). In some cases the participants began "from scratch", starting with an empty segmentation; in other cases, the participants edited an existing segmentation[3] produced by an algorithm[4]. Each participant was asked to segment the page according to their own interpretation of the page structure and at the "natural" level of detail; one of the most interesting aspects of this study is determining what the natural structure and level of detail are, and whether different participants perceive these differently. There were considerable differences in the time taken to complete each segmentation; as a result, not all participants were able to segment all pages in a session of approximately 75 minutes.

We developed a customized tool for participants to use to create and edit page segmentation trees. Based on the properties of the domain, as described in Chapter 2, we designed the interface to be based on axis-parallel, rectangular regions. This considerably simplified the required editing operations relative to a tool capable of editing general connected regions, and made the tool easier for participants to learn to use. The available operations were as follows: **Create** a new region; **Select** an existing region; **Delete** the selected region; **Resize** the selected region; **Move** the selected region; and **Zoom** in or out on the page. Figure 6.1 shows a partial screenshot of the interface, depicting the toolbar (to the right of the page) and a partially segmented page. To allow flexibility in deployment, the segmentation tool was designed to run in a web browser.

All editing operations included snapping to nearby edges, helpful in allowing users to produce a dense segmentation of abutting regions without requiring pixel-accurate positioning. (If, after an editing operation, a vertical edge of the edited region was within 10 pixels horizontally of a snap line, the edge was moved to coincide with the snap line; horizontal edges were treated analogously. Each image boundary was considered a snap line, and snap lines were created for each edge of a region, extending 20 pixels beyond the region itself to allow for alignment between regions that have whitespace between them.) The hierarchy of the segmentation tree was automatically inferred from containment relationships: a region was a descendent of every region that contained it, and a child of the

[2]This included undergraduate and graduate students at the University of Waterloo, as well as others obtained by word of mouth.

[3]Figure 6.1 shows several regions (outlined with blue horizontal lines and green vertical lines) as they would be seen by the participant if this was the initial segmentation.

[4]The algorithm used for this purpose was the improved segmentation algorithm described in Section 3.3

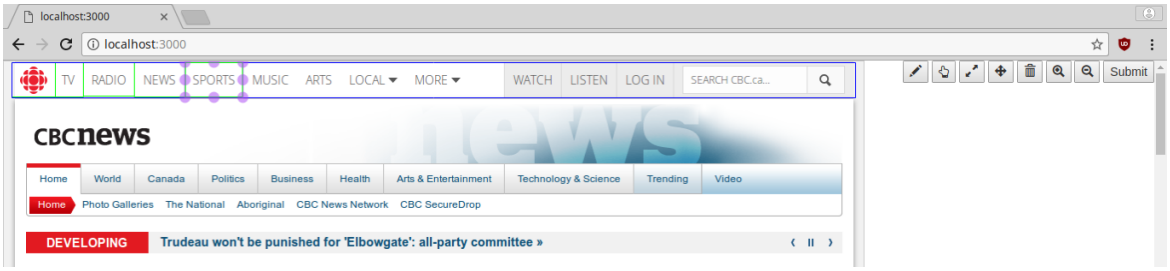


Figure 6.1: Screenshot (cropped to save space) of the segmentation editing tool used in this study. The toolbar is show to the right of the page, and several regions have been created. In this example, the server is being run locally.

smallest such region. When two regions overlapped without a containment relationship, the overlapping area was highlighted; if the user attempted to submit without resolving this, an error was raised. The hierarchical structure of the segmentation tree was represented to the user by colouring the outlines of regions according to their depth in the hierarchy; this helped to clarify ambiguous cases where region edges met.

Page selection was an important consideration. Because the objective of the study was to produce large numbers of segmentations for each page in order to study segmentation properties across a range of users, it was necessary to restrict the number of pages used to five. The pages chosen were: (1) `bbc_arts`, the arts page of BBC News; (2) `bbc_health`, the health page of BBC news; (3) `bbc_main`, the front page of BBC News; (4) `cbc_main`, the front page of CBC News; and (5) `nyt_politics`, the politics page of the New York Times. News web pages were chosen because they are complex enough to produce interesting segmentations, popular enough to represent a major area of web browsing, and similar but not identical in structure, to allow useful comparisons across pages in a small dataset. These examples were selected from a larger dataset used in qualitative evaluation of a segmentation algorithm [34].

Had we used a larger set of pages, it would have been necessary to reduce the number of participants segmenting each page. This, however, would have compromised our ability to draw conclusions about the degree of consistency between users' segmentations. By studying a small number of pages but obtaining more segmentations of each, we are able to draw conclusions about consistency between different users' segmentations of a given page and about common trends in segmentations across a user's segmentations of different pages. This, in turn, provides insight into the appropriate methods for collecting segmentations, the number of segmentations required for each page in a larger dataset, and the

interpretation of segmentation quality results obtained from a single ground truth segmentation (especially what level of disagreement can be expected between human users, whose segmentations must be assumed to be equally good).

To produce a large dataset for standardized evaluation of page segmentation algorithms, it will be necessary to change the methodology to obtain as few segmentations of each page as is consistent with the required quality of ground truth data, in order to maximize the number of distinct pages in the dataset; the requirements are different from those of the study presented here because the scientific objectives are different. One approach to simplifying the logistics of gathering a larger dataset would be to use a crowdsourcing platform such as Amazon Mechanical Turk (AMT) to obtain segmentations, rather than meeting with participants in person. We discuss this possibility further in Section 8.3.1.

The results reported here are based on 29 participants, who completed between one and five segmentations each, all required to do at least one "from scratch". Sixteen performed all segmentations starting from scratch; the other 13 performed at least one segmentation by editing an existing segmentation. Each of the five pages was segmented from 12 to 21 times. This represents a substantial dataset in terms of the number of segmentations.

6.2 Results

The analysis of our results can be conveniently divided into three areas: the structural properties of segmentations, the relationships between segmentations, and the relationships between participants.

6.2.1 Segmentation Structure

It is possible to quantitatively analyse important features of the segmentations produced by users. Of particular interest is the complexity of segmentations and the level of granularity. Here we examine the number of levels, number of nodes, node sizes, and numbers of child nodes to obtain a composite view of the structures of ground truth segmentation trees produced by different users. Such a view can be used as a prior probability distribution over segmentation structures. This is particularly useful in the case of a Bayesian segmentation algorithm such as those described in Chapter 3, in which an explicit prior probability distribution can be readily included.

Table 6.1 summarizes descriptive statistics for three key region-level features of a segmentation: height, width, and number of child nodes. To produce these values, we com-

	Normal?	Log-Normal?	Mean	Variance	Skew	Kurtosis
Region height	No ($\alpha = 0.001$)	No ($\alpha = 0.001$)	120.7	5.976×10^4	12.06	201.6
Region width	No ($\alpha = 0.001$)	No ($\alpha = 0.001$)	329.7	7.136×10^4	1.478	4.174
# of child regions	No ($\alpha = 0.001$)	N/A	0.8645	4.277	4.345	31.95

Table 6.1: Table of descriptive statistics for key structural features of ground truth segmentations drawn from scratch by users. Tests of normality and log-normality use the Lilliefors test [87].

bined data across all users and all pages for those segmentations produced from scratch, rather than by editing an initial segmentation. The tests of normality and log-normality used the Lilliefors test [87], because the parameters of the best-fit distributions are not known *a prioiri*. Even these summary statistics represent useful information about the expected structural properties of a segmentation that could be used in segmentation algorithms.

Figures 6.2 and 6.3 show the empirical distributions of these structural parameters as histograms. Several interesting qualitative features are visible in these graphs. The distributions of heights and widths obviously differ; small heights (*e.g.* under 100 pixels) are much more common than large heights, while the distribution of widths has a mode around 250 pixels and a second, smaller mode around 1000 pixels. The smaller peak in widths corresponds to the width of the page, and is probably associated with the tendency for headers and their major divisions to cross the full width of the page. The distribution of child regions also has interesting features. Most regions (8113 of 10843) are leaves, with no children. This is reflected in the small average number of children shown in Table 6.1. Of the regions that do have children, the distribution is skewed towards small numbers of child regions *(i.e., regions that do have children tend to have only a few).* These qualitative features could be incorporated into a segmentation algorithm in the form of tunable parameters, or as empirical distributions, represented parametrically or nonparametrically. This general approach can be readily applied in other domains, especially domains in which restrictive assumptions about region structures can be made (as is the case here, where regions are constrained to be axis-parallel rectangles). We return to the discussion of possible extension of our work to other domains in Section 8.3.3.

The average segmentation tree, over all pages, has 197 nodes, with a minimum of 11 and a maximum of 596. In the 500-image Berkeley Segmentation Dataset (BSDS500) [8]) used for natural images, the average number of nodes per segmentation is only 20.9, with a minimum of 2 and a maximum of 208; our web page segmentations tend to be much

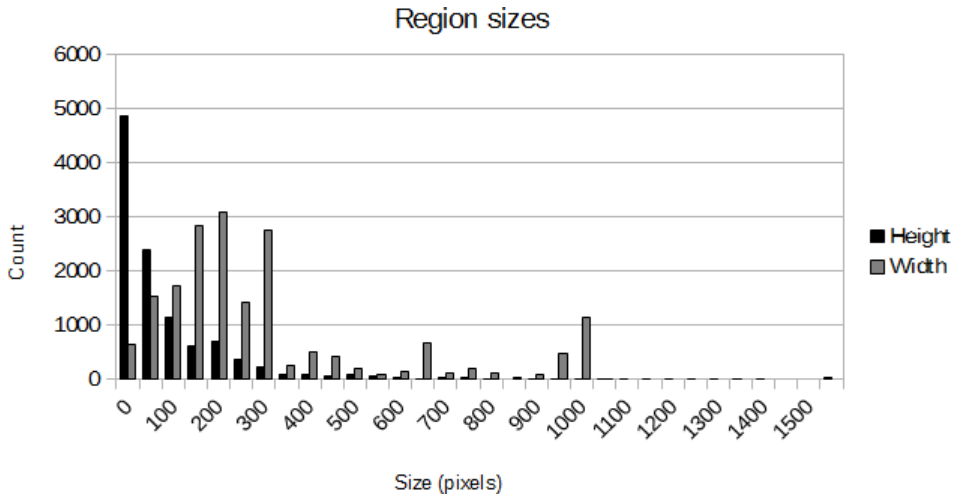


Figure 6.2: Heights and widths of regions, across all pages and all segmentations produced from scratch.

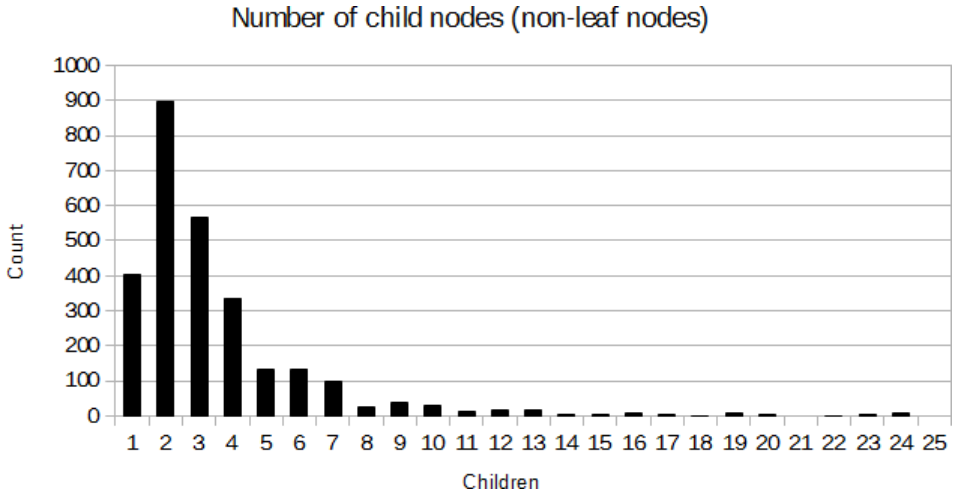


Figure 6.3: Numbers of children for all non-leaf regions, across all pages and all segmentations produced from scratch. The total number of leaf regions (with 0 children) is 8113, much larger than any other number of child regions; this bar is omitted here for clarity.

more complex, but we are able to operate well in this environment. We also use more participants per page, an average of 15.4 as opposed to 5.39, which increases the chance that we are working with a representative sample of segmentations[5].

Comparing the structures of the segmentation trees produced from scratch to those produced by editing an existing segmentation produces informative results. Figures 6.4 and 6.5 show the cumulative distribution functions of the numbers of nodes and numbers of layers, including all pages. In both cases, the differences between the two distributions are statistically significant, for $\alpha = 0.01$ in the first case and for $\alpha = 0.001$ in the second, using the two-sample Kolmogorov-Smirnov test. The proportions of examples from each page are unfortunately not identical between the two conditions (due to the different numbers of pages segmented by different users), but there are no statistically significant differences between the distributions for different pages, as shown above, and for tree depth in particular the results show a very strong difference. Taking individual pairs of pages, the two conditions produce statistically significant results for `bbc_arts` with respect to the total number of nodes ($\alpha = 0.05$) and the tree depth ($\alpha = 0.01$); without combining pages, it is impossible to draw a statistically significant conclusion for `bbc_main` at the $\alpha = 0.05$ significance level, and all other pages have much higher significance thresholds than `bbc_arts`. Although limited by the small number of pages segmented by editing, these results indicate that the segmentations produced from scratch and from an initial segmentation differ in structural features for at least some pages. The difference in the way participants report their perception of the organization of the page raises an important question: what starting point produces the "most useful" segmentations? The answer may depend upon the intended use of the ground truth segmentation, but the question must be considered.

6.2.2 Comparison of Segmentations

The results of our experiments suggest a broad qualitative consistency between segmentations produced by different users and with different starting points. Figure 6.6 shows an example page (specifically `bbc_arts`) in greyscale, with outlines of all regions from all segmentations overlaid on it. Note that, at a large scale, the structures are quite consistent. Even in areas of agreement, individual users rarely agree down to the pixel; placement of regions and the amount of whitespace border included in a given region vary from one user to another. At smaller scales, additional interesting differences can be seen. Note, in

[5]The average number of participants per page is less than the total number of participants because not every participant completed all five pages in the time available.

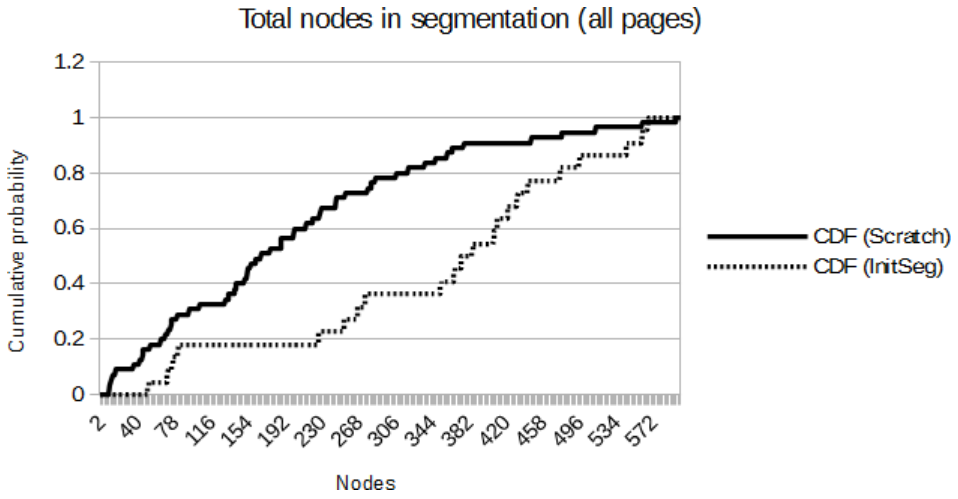


Figure 6.4: Total number of nodes per segmentation (cumulative distribution function) over all pages.

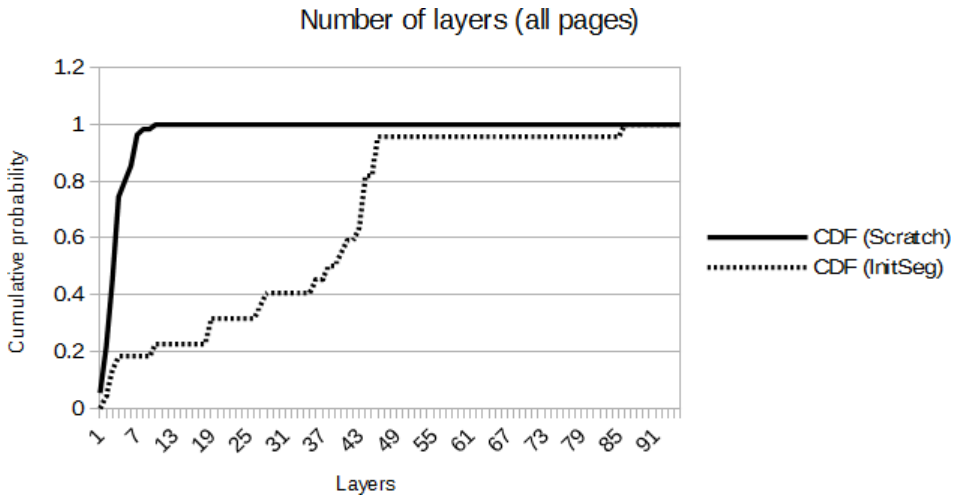


Figure 6.5: Total number of layers per segmentation (cumulative distribution function) over all pages.

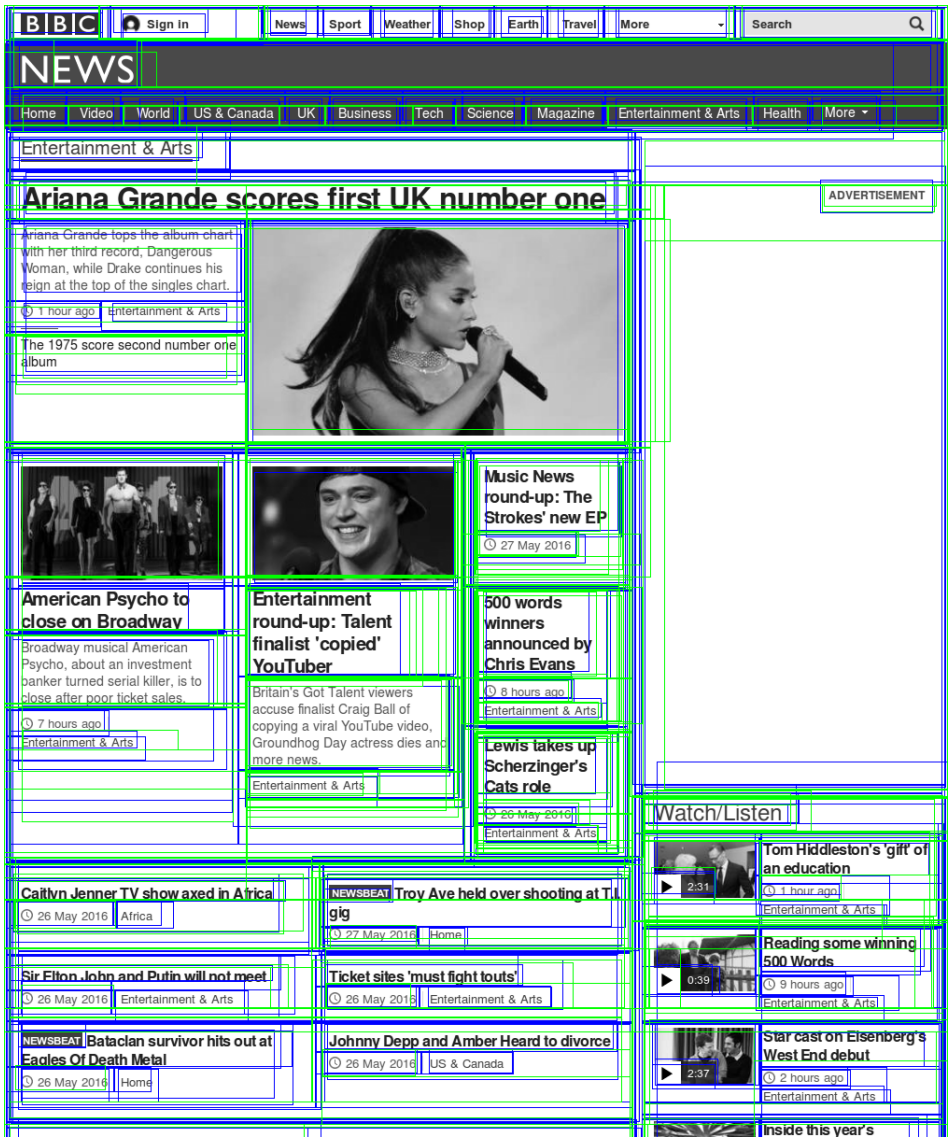


Figure 6.6: Excerpt from the page `bbc_arts`, showing all participants' segmentations in the same image. Segmentations produced from scratch are shown in blue; those produced by editing an initial segmentation are shown in green.

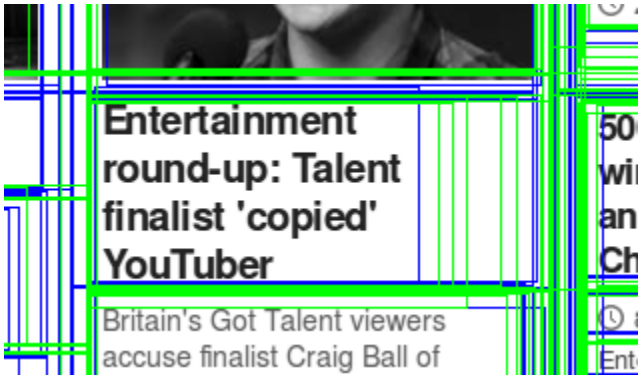


Figure 6.7: Detail view of a small area of Figure 6.6, showing differences between segmentations.

Figure 6.7, that users differ in the compromise they make between matching the edge of the headline and matching the thumbnail above it. The results here demonstrate that there is a ceiling on the performance (measured by agreement with a ground truth segmentation) of any single segmentation algorithm; perfect agreement with one person's segmentation implies some disagreement with others, all of which are equally correct. For minor differences this is not significant; for more significant structural differences, it may be necessary to personalize segmentation algorithms to suit individual users (*e.g.* for assistive needs).

These qualitative results suggest that there is a performance ceiling for segmentation algorithms when using a simple measurement of agreement with any single ground truth segmentation. Because human participants do not agree perfectly, it is impossible to define a unique "perfect" segmentation. These same results do, however, suggest that "good" segmentations could agree well with segmentations provided by many users at a large scale; an algorithm that produces such a segmentation could be considered a compromise between the needs of many different users. It may also be possible to adapt a compromise algorithm to the preferences of a group of users or even a of a single user, in order to more closely match their preferences in the details of segmentations. We return to this idea in Section 6.2.3.

In order to quantitatively compare segmentations, it is necessary to define a measurement of similarity. In Section 3.2.8 the earth mover's distance [101, 102] between the outlines of a pair of segmentations was used to compute their similarity. In this method, the distance is defined to be the "effort" required to transform one distribution into another, considering both the amount of "mass" transferred between variables and the distance it

is moved. This approach has many useful properties, but it is slow to compute (requiring downsampling of the segmentations to be feasible) and the pixel-level granularity does not enforce one-to-one associations between nodes in the two segmentations. For the experiments described here, we use a node-level edit distance to define similarity; as a result, we are able to explicitly associate between optimal pairs of nodes, and compute exact edit distances between complete segmentations at full scale.

For any two sets P and Q of regions such that $|P| = |Q|$, we define the edit distance between the two sets to be

$$D_{set}(P, Q) = \min_{f} \left(\sum_{p_i \in P} D_{node}(p_i, f(p_i)) \right) \tag{6.1}$$

for some association $f : P \to Q$ between the two sets. In other words, it is the cost of the minimum weighted bipartite graph match between P and Q, with the weights defined by a distance function D_{node}, which represents the edit distance between a pair of individual nodes. In general, of course, the number of regions in a pair of segmentations is not necessarily equal, and not every node in one segmentation will have an equivalent in the other. To account for this, we augment the sets of regions with placeholder nodes. Thus, for two segmentations $S_1 = (P, E_P)$ and $S_2 = (Q, E_Q)$, the edit distance between the two *segmentations* is $D_{seg}(S_1, S_2) = D_{set}(P \cup R_1, Q \cup R_2)$, where $R_1 = \{r_{1,1} \cdots r_{1,|Q|}\}$ and $R_2 = \{r_{2,1} \cdots r_{2,|P|}\}$ are sets of placeholder nodes. We define the edit distance between two *nodes* to be

$$D_{node}(p, q) = \begin{cases} \sum_{s \in \{l,r,t,b\}} p_s - q_s & p, q \text{ regions;} \\ \text{width}(p) + \text{height}(p) & p \text{ a region,} \\ & q \text{ a placeholder;} \\ \text{width}(q) + \text{height}(q) & p \text{ a placeholder,} \\ & q \text{ a region;} \\ 0 & p, q \text{ placeholders} \end{cases} \tag{6.2}$$

where p_l, p_r, p_t, and p_b are the positions of the left, right, top, and bottom sides of p, respectively. This reflects the difference between two regions and the size of regions "created" by matching with a placeholder; unused placeholders do not affect the edit distance between two segmentations[6].

Using this definition of similarity between regions, we computed pairwise similarity for each pair of segmentations produced from scratch for each page. Figure 6.8 shows the cumulative distribution functions of edit distances over all pages for the distances between pairs of segmentations of a page performed from scratch, pairs of segmentations performed

[6]The placeholders are only used to allow non-matching regions with a fixed problem size; matching between two placeholders "discards" both with zero cost

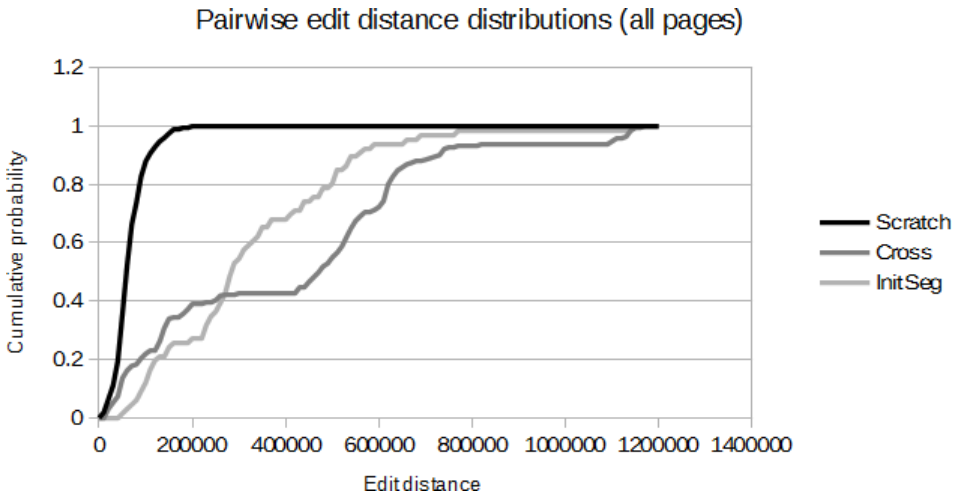


Figure 6.8: Cumulative distribution functions of edit distances across pairs of segmentations of the same image. Scratch means both segmentations produced from scratch; InitSeg means both segmentations produced by editing an initial segmentation; Cross means one segmentation produced by each method.

by editing, and pairs of segmentations produced by different methods. Data from all pages has been combined to produce the distributions shown. Figure 6.9 shows the same measurement considering only the top 3 levels of the segmentation tree. In each case, the difference between pairs of distributions is statistically significant for at least $\alpha = 0.05$ by the Kolmogorov-Smirnov two-sample test. Note the high degree of agreement in the first three levels of the tree when the segmentations were produced by editing; a significant number of these segmentations are identical or nearly so. For complete segmentations, the level of agreement is highest for segmentations produced from scratch, possibly due to differences in how closely the users stick to the starting segmentation.

6.2.3 Comparison of Participants

To compare participants and determine how similar different participants' segmentation structures were, we examined the distributions of structural features examined in Section 6.2.1 (region height, region width, and number of child regions) and compared them for statistically significant differences using the two-sample Kolmogorov-Smirnov test. Figure 6.10 shows matrices representing pairs of participants whose distributions are not statistically different for each feature. Most participants are indistinguishable from some others

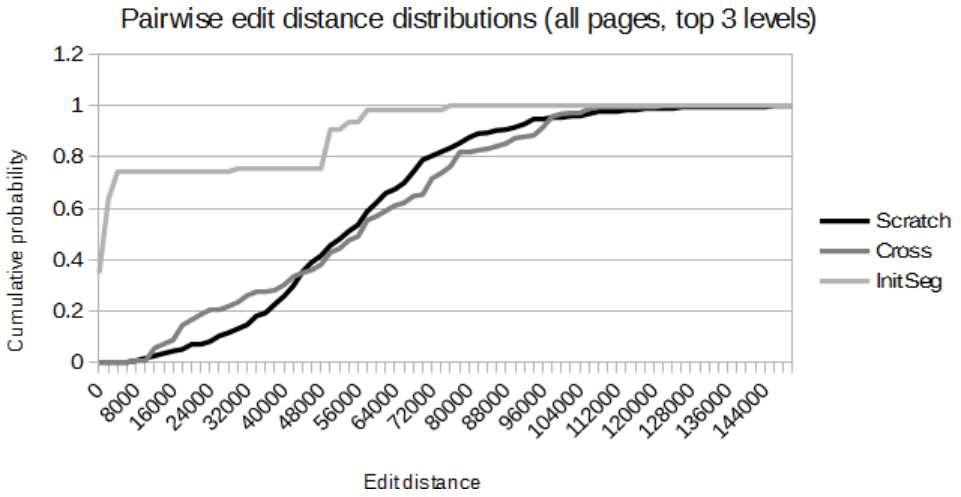


Figure 6.9: Cumulative distribution functions of edit distances across pairs of segmentations of the same image. Data series defined as in Figure 6.8.

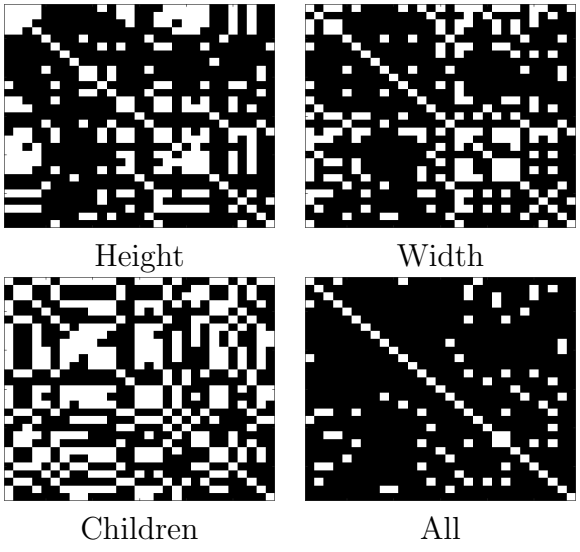


Figure 6.10: Matrices showing users with statistically indistinguishable distributions for each of the shown structural characteristics ("all" means none of the three distributions are individually statistically distinguishable). White indicates a match between participants; note that this does occur in "All".

in one respect or another, and 25 of 29 participants match at least one other participant in at least one respect. This suggests that despite significant differences there are also significant similarities between users.

6.3 Discussion

Our results demonstrate that there is value in studying many segmentations of each image in a small dataset, since we have thereby obtained valuable insights into the segmentation problem for images of web pages. Additionally, we establish valuable structural characteristics for a prior probability distribution over segmentations in our domain (images of web pages). We have also introduced a method for eliciting ground truth by editing existing segmentations using our editing tool, and demonstrated the capabilities and limitations of this approach.

Our results suggest that there is a ceiling for the performance of "one-size-fits-all" segmentation algorithms when compared with individual or even consensus ground truth segmentations in this domain due to disagreements between participants about what constitutes a "correct" segmentation. One important avenue for further research is the possibility of adapting segmentation algorithms to individual user preferences. The same statistical properties of segmentation studied in Section 6.2.1 to produce a prior over web page segmentation trees can be studied for individual users, as in Section 6.2.3, thus providing individualized segmentation priors. The same principle could be applied to other computer vision systems, raising the exciting possibility of personalized output for users. We plan to continue to examine these possibilities in future work (see Section 8.3.5).

Perhaps the most significant generalizable contribution from our work is the analysis of the role of the initialization of the segmentation when eliciting ground truth segmentations from participants. An empty initial segmentation (*i.e.* "starting from scratch") is an obvious choice, but non-empty initializations may prove useful in some circumstances, including in evaluating the algorithm used to create the initial segmentation. Our results clearly show that the resulting segmentations can differ; they leave open the question of what initialization method should be preferred as one for consideration in the design of each user study. In all, we provide new directions for eliciting ground truth segmentations in computer vision.

6.3.1 Evaluating Segmentation Quality

One of the primary motivations for conducting this study was to generate ground truth for use in evaluating our segmentation algorithm (Segment) from Chapter 3. The idea was that we would run our segmentation algorithm on the same dataset as the one employed in the user study. We could then compare the segmentations produced by our algorithm to the ground truth segmentations produced by participants on the basis of edit distance. A small edit distance suggests a high degree of agreement between our algorithm and the ground truth segmentations, which in turn indicates that our algorithm is performing well.

In Table 6.2, we show for each page the mean edit distance between our automatic segmentation and each ground truth segmentation produced by the study participants, as well as the average edit distance for all pages combined, distinguished by the starting point (*i.e.*, whether or not the participants used an automatic segmentation as a starting point). These edit distances are higher than those observed between ground truth segmentations produced from scratch, but are roughly comparable to those between ground truth segmentations produced starting from an initial segmentation that was produced by our algorithm (see Figure 6.8). This is a very promising indication of the quality of our segmentations, especially since the ground truth segmentations tend to be flatter than our automatic segmentations.

To explain in more detail, the mean edit distances for initialized segmentations (Init-Seg in Table 6.2) turn out to be comparable to the typical distances observed between users. It is important to note that *all* of the ground truth segmentations are in some sense correct, since they were provided by human participants as their own interpretation of the page structure. The ground truth segmentations can be considered as points in a cloud in a high-dimensional space or manifold with distances defined by the edit distance between them; the average distance from our segmentation to points in this cloud then turns out to be roughly comparable to the median distance between points in this cloud. For segmentations produced from scratch, the cloud is more compact, and the mean edit distance to points in this cloud is rather larger than the cloud itself; examining Figure 6.5, we can see that these segmentations have fewer layers than those produced by editing, which may explain the greater absolute and relative distance in this case. Overall, these results indicate that our segmentation algorithm performs well on this dataset, producing good-quality segmentations which agree well with those produced as ground truth segmentations by participants. In particular, the difference between our automatic segmentation and those produced by participants as ground truth segmentations is comparable to the difference between different ground truth segmentations when the participants started with an automatic segmentation. For segmentations produced from scratch, the average edit

	bbc_main	bbc_arts	bbc_health	cbc_main	nyt_politics	Total
Scratch	6.61 ± 0.08	3.73 ± 0.11	4.01 ± 0.09	4.49 ± 0.10	6.62 ± 0.07	5.52 ± 0.17
InitSeg	5.97 ± 0.19	2.55 ± 0.19	3.17 ± 0.58	2.94 ± 0.71	6.20 ± 0.51	3.35 ± 0.33

Table 6.2: Mean edit distances between our segmentation and each ground truth segmentation for each page and for all pages combined. All values $\times 10^5$.

distance to our automatic segmentation is slightly higher, and the edit distances between participants are smaller. The performance of our segmentation algorithm against both parts of the dataset suggests performance approaching that which would be observed if one person produced segmentations manually for evaluation by others.

It is also interesting to note that the mean edit distance between our automatic segmentation and ground truth segmentations produced by editing an automatic segmentation (InitSeg) is smaller than that between our automatic segmentation and ground truth segmentations produced from scratch; this difference is significant for $\alpha = 0.001$. This demonstrates that the starting point for segmentation editing does influence the outcome of the evaluation process, and therefore supports our assertion that the starting point is an important consideration for eliciting segmentations for a test dataset.

Chapter 7

Discussion and Related Work

In this chapter we describe related work in the key areas that our research touches upon. We contrast the computer vision techniques used in our segmentation-classification pipeline with related techniques from the corresponding areas in computer vision to illustrate the contributions made to computer vision in the course of this research. We also contrast our approach to vision-based web page parsing with other methods of web page parsing to demonstrate the novelty of our solution and its advantages in this application domain. Finally, we discuss the state of assistive technology on the web, to demonstrate the need for a vision-based web page parsing system such as ours.

We have already drawn out primary comparisons with related work in computer vision in Section 2.5. Appendix C provides a deeper exposition of that related work.

7.1 Simultaneous Segmentation and Classification

Segmentation-classification pipelines are one approach to the problem of parsing an image. As noted in Section C.3, this is a rather popular approach for depth images and three-dimensional point clouds. Another approach is to perform segmentation and classification simultaneously. This approach is very commonly used in parsing natural images. Simultaneous segmentation and classification allows the system to leverage class-specific cues about objects in the image to aid in the segmentation process. An important disadvantage is that only objects with known characteristics can take advantage of the combination of segmentation and classification. In many cases, a generic "background" region is included to encompass all unrecognized areas in the image. This region can combine multiple semantically and visually different regions, resulting in a segmentation that is in some sense

(a) Semantic segmentation

(b) Instance segmentation

Figure 7.1: Examples of semantic and instance segmentation, performed manually, on a natural image. Semantic segmentation, left, produces two foreground regions (a "book" class in blue and a "moulding plane" class in red). Instance segmentation divides separate instances of the instance class, resulting in four segments, three of which area classified as "moulding planes".

incomplete; for many applications, however, the details of the background are irrelevant, and it is only specific foreground objects (such as cars, pedestrians, and street signs) that are of interest. Algorithms that combine segmentation and classification into a single step can be divided into two classes: semantic segmentation, which creates a single (possibly non-contiguous) region for each class, and instance segmentation, which creates separate regions for each object in each class.

Figure 7.1 shows an example of manual segmentation of a photograph following both the semantic segmentation (7.1a) and instance segmentation (7.1b) models. In both cases, the tools (a group of three antique moulding planes) are separated from the book, the table, and the background. For semantic segmentation, however, the three planes form a single region, while for instance segmentation they are separated into three examples. Distinguishing between instances is often important; continuing from this example, a system intended to assist in finding a rare plane worth two thousand dollars among hundreds of common twenty-dollar planes must accurately distinguish between instances in order to identify stylistic cues to the age and origin of each plane.

7.1.1 Semantic Segmentation

Semantic segmentation is the process of classifying pixels in the image to produce a map of semantic classes [51]. For an image of a street scene, for example, each pixel may be classified as "building", "road", "vehicle", "pedestrian", "tree", or "background". This map implies a segmentation with one region for each class. The problem of semantic segmentation is distinct from the problem of object detection; an object detector is designed to find the bounding boxes of instances of a specific class of object in the image. It uses a single class (although the class may be quite general and the output from multiple object detectors can be combined), and the output is only a bounding box, not the exact boundaries of the object. It is important to note that semantic segmentation is not an appropriate method for the intended applications of our web page parsing system because it does not differentiate between instances of a class; instance segmentation, described below, is required for this. As such, we discuss semantic segmentation only briefly here, although it is a common technique in the domain of natural images.

The problem of semantic segmentation has been approached in many different ways. One paradigm that has seen extensive use is conditional random field (CRF) models[1] of various types. An illustrative example of the use of CRFs in a semantic segmentation is the multiscale CRF model proposed by He *et al.* [63]. This approach uses unary potentials proportional to the probability of a specific label at a given pixel given the image data, pairwise potentials proportional to the probability of a pair of pixels given the image data, and higher-order "label feature" potentials proportional to the probability of a labelling of a larger set of pixels given the image data. These higher-order label features are defined at multiple scales (hence the term "multiscale CRF"), including a global scale which takes advantage of properties of image composition, as the eigenregion model of Fredembach *et al.* did [49]. This model allowed the enforcement of spatial coherence as well as large-scale constraints on plausible image interpretations.

Another approach to semantic segmentation—one which is currently very common—is the use of convolutional neural networks and other deep learning methods. A remarkable range of architectures have been proposed for this area. It is beyond the scope of this chapter to provide a detailed survey of the techniques used in this area, since our work uses neither semantic segmentation nor a convolutional neural network architecture. A 2017 review article by Garcia-Garcia *et al.* [51] describes major deep learning methods for semantic segmentation, including the use of CRF-based postprocessing algorithms to improve spatial resolution.

[1] CRFs are discussed in Section C.2.3 of this thesis.

7.1.2 Instance Segmentation

Unlike semantic segmentation, which can be viewed as per-pixel classification, the objective of instance segmentation is to separate not only pixels of different classes, but also pixels in different instances of the same class. Distinguishing between instances of a given class is very important for many areas. A traffic camera being used to measure traffic levels, for example, must count instances, and many instances may be in the field of view at once; obviously it is impossible to count instances accurately if the instances cannot be separated from each other. If the image is a rendered web page, it is likely that similar regions (such as article blurbs) will be adjacent to each other. With a semantic segmentation algorithm, a column consisting of article blurbs will not be divided; with instance segmentation, the column will be divided into individual blurbs which can then be individually processed.

Instance segmentation is in some respects more challenging than semantic segmentation. Not only must the type of object be accurately recognized for each pixel, the boundaries between objects of the same type must be accurately located. On the other hand, as pointed out by Yang *et al.* [135], template-based models of an object can be of more use for a single object instance than for a region consisting of multiple objects. A wide variety of methods have been proposed for instance segmentation. As with semantic segmentation, many modern methods are implemented using deep learning methods, but other approaches also exist.

A common method for instance segmentation is to refine the bounding boxes produced by an object detection algorithm to produce exact boundaries of classified objects. Arnab *et al.*, for example, used a combination of cues, including object detection bounding boxes and a semantic segmentation, and combined them using a CRF [10]. Li *et al.* proposed an algorithm that begins with a bounding box from a detector and uses a convolutional neural network to iteratively refine the object boundaries; Li *et al.* argued that repeated local corrections to the segmentation are easier to model than global corrections [86].

Yang *et al.* proposed in 2012 an interesting Bayesian instance segmentation method [135]. This segmentation algorithm uses initial region proposals provided by object detection and refines these detections to produce an instance segmentation, accounting for depth ordering and object shape. These elements are modelled in a common Bayesian framework, and the system combines top-down and bottom-up cues. Because this algorithm takes a probabilistic approach broadly similar to our own, it is worth considering in detail. While our algorithms were defined in Sections 3.2 and 3.3 in bottom-up order, beginning with the definition of the probability of that there is a locally significant edge at a given pixel and finishing with the definition of the probability of an entire segmentation, we will follow the top-down order used in [135] in describing Yang *et al.*'s algorithm.

In order to discuss the mathematical details of the segmentation algorithm in question, it is necessary to first define some notation. The notation in the following discussion is based closely on the definition of the algorithm in [135] with slight modifications to ensure clarity in this brief discussion without derivations, and to avoid conflicts with other notation in this thesis. Let d_n, for $n \in \{1 \cdots N\}$, represent the n^{th} detection, including its class, position, and score. Each detection corresponds to exactly one depth level, and vice-versa. The layers are ordered according to a permutation π; $d_{\pi(N)}$ is the front detection, and $d_{\pi(1)}$ is the back detection; $d_{\pi(0)}$ represents a "background" region, corresponding to a "detection" of the entire image. Each layer has a corresponding set of parameters for the appearance model, denoted θ_n for the $\pi(n)^{\text{th}}$ detection. Each region has a corresponding segmentation mask which refines the bounding box to the outline of the object; b_{im} is a binary variable such that $b_{im} = 1$ if and only if the pixel i is part of the object detected in layer m. The feature value at pixel i is denoted x_i. The final label of pixel i is denoted x_i, where $x_i \in \{0 \cdots N\}$; due to occlusion, each pixel is labelled with exactly one layer, although it may be part of many occluded objects.

The labels and features of different pixels are assumed to be independent, so the probability of the feature values x and an assignment z of labels to all pixels, given by the appearance parameters θ, detections d, and depth ordering π is defined to be

$$\Pr(z, x|\theta, d_\pi) = \prod_{i=1}^{N} \Pr(z_i, x_i|\theta, d_\pi) \tag{7.1}$$

Because the probability of a label is dependent only on the labelling and the probability of a feature value is dependent only upon the appearance model, this equation can be factored again:

$$\Pr(z, x|\theta, d_\pi) = \prod_{i=1}^{N} \Pr(z_i|d_\pi)\Pr(x_i|\theta_{z_i}) \tag{7.2}$$

The probability $\Pr(x_i|\theta_{z_i})$ allows the algorithm to model specific instances of a class; ignoring this term was found to significantly reduce performance. The probability $\Pr(z_i = m|d_\pi)$ requires further analysis to provide a satisfactory definition. Considering the physical interpretation of this probability is helpful. In effect, this is the probability that layer m is the furthest-forward layer with a segmentation mask including the pixel i. It can be shown, then, that this probability is equal to

$$\Pr(z_i = m|d_\pi) = \beta_{im} \prod_{n=m+1}^{N} (1 - \beta_{in}) \tag{7.3}$$

where $\beta_{in} = \Pr(b_{in} = 1|d_{\pi(n)})$. The value of β_{in} depends upon the class-specific shape model for layer n. In its simplest form, the shape model can be represented by a mask α for each class c. If the mask is a single model, $\beta_{in} = \alpha_{i',c_n}$, where c_n is the class of layer n and i' is the position of the pixel i in mask coordinates. Yang *et al.* also discuss the use of more sophisticated mixture- and part-based shape models; these are treated similarly, but have more ability to adapt to the image data. Using a mixture-of-deformable-parts model was found to help the performance of the model. In [135], the authors also discuss the use of superpixels to incorporate bottom-up grouping into the segmentation process; in the interest of brevity, we omit a detailed discussion of this aspect here, but it is worth noting that these bottom-up cues did help the overall performance of the algorithm.

The probability of a given pixel belonging to a given region depends upon the depth ordering of regions. One possible way to incorporate this is to simply assume a uniform probability distribution over all orderings. Yang *et al.* also present a method for estimating the probability of a given ordering based on the properties of the detections. Vertical position is useful in estimating the depth of objects that can be expected to lie on a common ground plane, height gives an indication of distance due to perspective effects, and objects that are partially occluded are likely to have a lower detection score than fully visible objects. The probability of an ordering π given a set d of detections is given by the conditional MRF distribution

$$\Pr(\pi|d) = \frac{1}{Z(d)} \prod_{m<n} \exp(-w^T(f_{\pi(m)} - f_{\pi(n)})) \tag{7.4}$$

where $f_{\pi(m)}$ represents a feature vector describing a detection, w is a learned model of depth cues, and $Z(d)$ is simply a normalizing constant. Layering was found to have only a slight impact in tests using the PASCAL VOC 2009 [44] and 2010 [45] segmentation datasets; Yang *et al.* suggest that this is due to the relatively sparse labelling present in this dataset leading to relatively few occlusions. The layering model is used to account for occlusions, and is therefore of little consequence if occlusions are rare. The impact of ordering was found to be greater on a subset with overlapping true positive detections.

The model described by Yang *et al.* for natural images is related to but distinct from our method for web page image segmentation. Both take a Bayesian approach to the segmentation problem, and incorporate specific constraints from the expected image formation process. In Yang *et al.*'s algorithm, these constraints assume an image formed by perspective projection from occluding opaque objects, ordered in depth. The rendering process by which an image of a web page is generated is much different, and as a result our assumptions differ from those of Yang *et al.* Our segmentation algorithm assumes hierarchically tiled regions in the form of axis-parallel rectangles. Occlusion is uncommon

in web pages (though it can occur if one region is overlaid on another), and is not modelled in our algorithm; similarly, perspective projection does not occur in the web page rendering process.

In addition to the differences in assumptions resulting from differences between the intended application areas, there are important conceptual differences between the two algorithms. The algorithm proposed by Yang *et al.* performs segmentation and classification simultaneously, and relies upon models of specific object classes in the segmentation process; this adds an important top-down component to the segmentation algorithm. Our algorithm, in contrast, is purely bottom-up as described in this work. All of the assumptions are derived directly from the image formation process, and class-specific models are not necessary. Similarly, our algorithm is designed to provide a complete segmentation of the page rather than a segmentation of specific foreground object instances, and does not use a generic "background" region for unknown objects. Our algorithm also performs only segmentation; we perform classification as a separate step.

7.1.3 Separate or Simultaneous?

In our work, we used separate segmentation and classification steps, rather than performing segmentation and classification simultaneously as in the case of semantic or instance segmentation. There are several reasons for taking this approach, in both theoretical and applied aspects of our work.

In our application domain, segmentation and classification have distinct uses. As shown in Section 3.4 and Chapter 5, a segmentation tree, without classification of regions, can support many important assistive features, including magnification and highlighting. For these applications, detecting visually coherent regions and their hierarchical relationships is often sufficient for practical purposes. A separate segmentation step allows these tasks to be disentangled from the challenges of classification. Furthermore, the classification problem, in our case, is extremely context-sensitive. Consider, for example, an image in the form of a line drawing. If it appears in the header of the page, it is probably a logo for the site or page; if, on the other hand, the same image appears embedded in the main article on the page, it is probably an illustration. Even if it is an illustration of a logo, it serves a different role in the page than the page logo does. By separating segmentation and classification, we can simultaneously optimize over all region labels in a fixed segmentation using a hidden Markov tree, allowing the context in which a region occurs to play a strong role in the classification process.

From a more theoretical perspective, separately studying segmentation and classifi-

cation methods is useful in elucidating the specific requirements for each of these tasks. Simultaneous segmentation and classification could compromise experimental control by combining two tasks with, possibly, very different requirements. While effective simultaneous segmentation and classification would demonstrate the efficacy of the vision-based approach, separating the two tasks facilitates a finer-grained analysis of the performance of the algorithms. There is, therefore, also a scientific reason to separate our classification and segmentation algorithms.

Having described the reasons why separate simultaneous segmentation and classification were appropriate for the purposes of this thesis, it is also worth noting that a simultaneous or iterative approach may have merit in this domain. Because we have studied the properties of the two aspects of the page parsing problem separately and arrived at an understanding of these problems' individual requirements, it becomes possible to take a principled approach to combining the two. One possible method for combining segmentation and classification would be to consider not only the probability of the segmentation tree when optimizing over segmentations, but also optimizing the probability of the optimal assignment of region labels to the segmentation tree. Although probably more complex to optimize, this would provide richer information to the segmentation optimization process, and would ensure that the segmentation tree is "compatible" with the classification model.

7.2 Document Segmentation

One common case of document structure analysis occurs in optical character recognition (OCR) technology. It is necessary for OCR to find the regions of text in the image. In this case, the objective is to find blocks of texts to be read by other parts of the system. Note that our objective is to produce semantically significant groups, whether of text, photographs, icons, or other items. Segmentation into very fine units, such as lines of text in a paragraph, is neither necessary nor desirable for our purposes.

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 are as well some fundamental differences. Page segmentation methods in OCR—as in the system proposed by Cesarini *et al.* [26]—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 [28] 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.

Early OCR segmentation work generally used binary images with rules for building up blocks of text or image from groups of pixels. Shih *et al.*, for example, use a method based on bridging small horizontal and vertical gaps, then finding connected components [118]. Zlatopolsky described a more complex segmentation system which used different heuristics for different types of merging and different types of object (*e.g.* line segments and text blocks) [142].

A recent survey of document segmentation (for OCR and other applications) [76] classifies work in this area into methods based on projection, on smearing, on connected components, and on analysis of the background. In projection, pixels are summed along a line of sight (generally horizontally or vertically relative to the document axes). Anomalous peaks or troughs in the resulting one-dimensional projections are presumed to be caused by components in the document [76]. Smearing methods attempt to expand the foreground to fill each component. Methods based on morphological operations are a generalization of smearing methods [76]. One notable method using connected components is the construction of a graph of connected components with edges weighted by the Euclidean distance between component centroids; subtrees of the minimum spanning tree of this graph are considered to be page regions [76]. Methods using the background to define components rather than the foreground include the white tiles method, which finds areas of whitespace, the edges of which are used as separators [6]. All of these methods are distinct from our segmentation method, which is based on probabilistically finding edges and alignments of edges.

Document segmentation can also refer to the process of segmenting electronic documents (rather than scans of physical documents), especially when using a method which is applicable to multiple document types. Such a system was proposed by Burget [21], and later refined for web documents by Burget and Rudolfova [20]. This method is based on boxes and common visual properties that are applicable to a range of documents, including PDF, RTF, and OpenDocument formats as well as HTML.

The method proposed in [21] uses a two-stage process to produce its interpretation of the page. Page segmentation consists of the process of producing a tree of basic regions using a bottom-up algorithm. First, a tree consisting of all boxes in the representation of the document, plus a "virtual" box consisting of the entire page, is produced; containment relationships define the hierarchy (as in our work in Chapter 6), and when regions overlap the box that is "further back" is assumed to be the containing box. Boxes which do not

contain text and are not otherwise distinguished from their neighbours are pruned from the tree, and contiguous areas of boxes which are not visually distinct are merged. The segmentation tree is completed by finding maximal bounding boxes which do not cross visual separators. The second step is the detection of logical structure using font and punctuation evidence. A series of heuristic rules are used to estimate the level in the logical hierarchy of each image. This allows sibling nodes in the initial segmentation tree to be placed in different levels in the logical structure of the document.

Cesarini *et al.* described a tree structure designed to represent the series of horizontal and vertical divisions found in a typical document; they call this structure the modified X-Y tree [26]. A standard X-Y tree is built from the top down by recursively dividing detected regions; regions are divided through whitespace or lines, and the direction of division (horizontal or vertical) alternates at successive levels in the tree (a node with exactly one child can be used to represent consecutive divisions along the same axis). Thresholds on the dimensions of lines and whitespace regions are used to avoid dividing at meaningless positions. In a modified X-Y tree, each node is associated with features describing its type and coordinates in the document image, and additional edges between nodes in the tree are used to represent adjacency relationships. This is a richer model, although like the standard X-Y tree [96] it requires that separators (*i.e.*, the lines that divide regions in the segmentation) extend across an entire node. Both the modified X-Y tree and the standard X-Y tree used in our work reflect assumptions about document structure that are common to a range of domains, including printed material and web pages.

7.3 Web Page Segmentation

Visual features have been used to segment web pages for over a decade. A useful overview of some of the different efforts in doing so is offered in [61], which clarifies that vision-based approaches segment the web page from the browser-side perspective as it is rendered. Most of the existing web oriented segmentation methods use some combination of code features such as HTML tags and visual features derived from the source code or DOM tree such as background colour and font size. Our image-based approach does not use source code derived features directly, but only through their effect on an image of the rendered page. A large group of methods use some kind of vector representation of the rendered page; although they work with the rendered content elements and their visual features, they obtain these features from the DOM tree rather than from an image of the resulting page.

In 2003, Cai et al. proposed the Visual Page Segmentation (VIPS) algorithm [22], which uses cues obtained from the DOM tree to segment regions in the page. Like our

segmentation algorithm, VIPS takes a top-down approach, segmenting first the entire page and then recursively segmenting the resulting regions into smaller regions. The VIPS algorithm segments a region in three stages: extraction of blocks, detection of separators between blocks, and structure extraction. Block extraction and separator detection is a splitting process; the extraction of the structure of the region is a merging process.

VIPS divides DOM tree nodes into several categories according to their roles in defining the page. One important dichotomy, based on the HTML 4.01 specification, is the division into *inline nodes*, which can be applied to text without causing a line break, and *line-break nodes*, which always introduce a line break. VIPS also defines additional classes of nodes, which may overlap. A *valid node* is a node that is visible in the page, with nonzero height and width. A *text node* is any node in the DOM tree corresponding to text; these nodes do not have an HTML tag, although their parent node might. A *virtual text node* is any inline node with only text node children, or with only text node and virtual text node children.

A block, in the terminology of Cai *et al.*, consists of a set of nodes in the DOM tree. Each block is assigned a degree of coherence (DoC) score, based on a series of heuristic rules. To divide a parent block into its child blocks, the children of the DOM tree nodes corresponding to the parent block are examined to determine if they are sufficiently coherent to represent a single block or if they should be divided. When identifying blocks at this stage, a total of thirteen heuristic rules are used; these rules include "If the DOM node is not a text node and it has no valid children, then this node cannot be divided and will be cut", "If one of the child nodes of the DOM node is line-break node, then divide this DOM node", and "If one of the child nodes of this DOM node has HTML tag `<HR>`, then divide this DOM node" (rules quoted verbatim from [22]). Other rules set a DoC value, rather than requiring a node to be divided or not divided. Child nodes of dividable nodes in the DOM tree are recursively examined until a set of child blocks each corresponding to a single node in the DOM tree has been obtained.

Once a set of child blocks has been obtained, separators between these blocks are found. A separator, in the sense of [22], is a horizontal or vertical line across a region which does not cross any of the blocks in the page. Separators are characterized by their adjacent blocks and by their width. The weight of a separator is set according to a series of heuristic rules:

- Wider separators have a higher weight than narrow separators
- If the separator includes an HTML tag such as `<HR>`, its weight is increased
- It the separator divides blocks with different background colours, its weight is increased

- A difference in font size across a horizontal separator increases its weight, especially if the font size is larger above
- Similar block structures on either side of a horizontal separator result in a reduced weight

Having extracted a set of blocks corresponding to single DOM tree nodes, and determined the weights of the spacers between these blocks, the VIPS algorithm constructs the final set of child blocks according to the structure of the page. Pairs of blocks divided by a separator with a weight less than a threshold value are merged into larger blocks. The resulting block is assigned a DoC based on the highest-weighted separator inside the block. The segmentation process is repeated on these final child blocks until no further divisions are necessary; this is typically determined by checking the DoC of each leaf block in the segmentation tree to determine if any are less than the minimum permitted degree of coherence (PDoC).

VIPS provides an interesting and illustrative point of comparison with our approach to the problem of web page parsing. VIPS uses many visual features, such as the relative positions of DOM tree nodes in the rendered page, background colour, and font sizes. Unlike in our algorithm, however, these features are extracted from the DOM tree, rather than from an image of the rendered page. This approach results in strong implementation dependence. It is incapable of dividing a page beyond the level of a single DOM tree node, so embedded content such as graphics cannot be parsed. The heuristics chosen are also dependent upon specific tags, which ties the VIPS algorithm strongly to HTML 4.01. The rules used to divide a block into child blocks prior to separator extraction are applied in special ways to `<TABLE>`, `<TR>`, `<TD>`, and `<P>` tags, and to inline text nodes; all other nodes are grouped together and treated identically. As a result, a page implemented in HTML5 that uses the many features and tags introduced in the HTML5 standard would have many nodes with tags that provide potentially valuable semantic information lumped together in a generic group. Our algorithm avoids implementation-dependence by avoiding the use of the implementation as evidence, instead using the rendered page as it was intended to be viewed by the designer. Furthermore, because the heuristic rules are hand-coded with hand-tuned thresholds and scores, rather than learned from data, updating VIPS is labour-intensive and prone to poor choices of parameters. Our Bayesian approach, in contrast, is nonparametric wherever this is feasible, uses no hand-coded heuristics, and has minimal parameters and thresholds requiring tuning. As a result, training is simple and principled, and our system can be easily re-trained if, despite its implementation independence, design patterns in the visual appearance of the page change sufficiently to require modification of the parsing method.

Many other projects have used the VIPS algorithm as a starting point, including work by Petasis *et al.* [103] and by Song *et al.* [120]. This work recognizes the importance of the visual properties of page regions in determining their position in the page structure, but it relies entirely on the DOM tree to produce a segmentation (along with heuristics in order to decide which elements actually create a visual segment). One important descendant of VIPS is the recent modernization described by Akpinar and Yesilada [2, 3]. Akpinar and Yesilada note that VIPS as proposed by Cai *et al.* cannot make use of HTML5 elements, which were created after the original VIPS algorithm, or dynamic content, which can cause "invalid" (invisible) nodes to become valid. Additionally, they point to deficiencies in the set of heuristic rules, due to lack of detail and insufficient use of certain style attributes such as margin and padding widths.

One important contribution of [3] is a modern, portable reference implementation of the modernized VIPS algorithm. The original VIPS algorithm required integration with an obsolete version of Internet Explorer, which is unable to correctly render many modern pages, and has compatibility problems with many modern systems. To obtain segmentations from VIPS for comparison with those produced by our algorithm (see Chapter 3), it was necessary to use a Windows XP SP3 virtual machine to run the VIPS reference implementation.

The quality of segmentations was evaluated by having participants rate the quality of segementations. Akpinar and Yesilada also studied segmentation preferences by rating the performance of five different levels of complexity [3]. The results suggest a statistically significant difference in preferred complexity between different demographic groups. Notably, the age and education level of participants affected complexity preferences, as did experience in web design. Over the entire group of participants, the highest level of complexity was preferred, and this remained consistent across pages with different levels of inherent complexity. The overall results indicated reasonably good segmentation performance; when comparing all five levels of complexity, the mean score was 3.64, treating a Likert scale as interval values between 1 and 5. Our tests of segmentation accuracy are not directly comparable, as we used ground truth segmentations. The use of ground truth segmentations allows a quantitative comparison with a known-good segmentation and allows much more detailed evaluation. Similarly, our experiments with segmentation preferences are considerably more fine-grained than those described in [3], since we compare manual segmentations rather than ratings of a small number of prepared segmentations. These labour-intensive methods did, however, require the use of somewhat smaller datasets; Akpinar and Yesilada used a 29-page dataset, while our ground truth dataset consisted of five pages, although with a large number of segmentations of each.

In 2017, Zeleny *et al.* described Box Clustering Segmentation (BCS), an interesting

segmentation algorithm based upon the visual properties of boxes produced by a rendering engine [139]. This algorithm produces a flat segmentation rather than a segmentation tree, as in many applications the leaf nodes are more important than the internal nodes. A set of nodes is extracted from the segmentation tree such that all text and image nodes are included, no other leaf nodes are included, and no boxes contain any others. In this set of boxes, those whose projections onto the x or y axis overlap are said to be semi-aligned.

In order to proceed with a description of BCS, it is necessary to define some terminology. The mutual position of a pair of boxes is defined to be "above" if and only if the two boxes are semi-aligned in the horizontal direction and one is above the other; "below", "left", and "right" are defined analogously, and if none of these conditions apply the two boxes have a mutual position of "other". The "absolute distance" between a pair of boxes is defined to be the distance between the two closest edges if the boxes are semi-aligned, and infinite otherwise. The direct neighbourhood of a box is the set of boxes which have a mutual position (which is not "other"), and are the closest box in that direction (as a result, the direct neighbourhood contains at most four boxes). The direct neighbourhood of a box is used to calculate the "relative distance" between two nodes, which is the arithmetic mean of the absolute distance between the nodes divided by the maximum distance in the direct neighbourhood of each.

In BCS, boxes can be grouped together into clusters based on similarity. The similarity measure between two nodes is calculated on the basis of relative distance, area, aspect ratio, and colour features. The similarity between a cluster and a box or between clusters is defined to be the mean similarity between boxes. In segmenting the page, similar regions are grouped together such that the similarity between two entities proposed for grouping is at least as high as a threshold similarity. This clustering threshold is a parameter which must be set carefully, and which has a significant impact on the quality and characteristics of the resulting segmentation. Each new candidate cluster is tested to ensure that it does not overlap with any other cluster or box; if a candidate cluster overlaps a box, it is still valid if all boxes which it overlaps can be added to the cluster. If a candidate cluster passes this test, it is accepted. The process of proposing and testing clusters then continues, ultimately producing a flat set of clusters which constitutes the segmentation.

BCS was evaluated using a dataset of 800 pages (100 in each of eight categories), each with three ground truth segmentations. These ground truth segmentations were prepared at the level of granularity of the DOM tree. Volunteers prepared one ground truth segmentation, and by associating the DOM tree nodes in that page with nodes in each other page in that category automatic segmentations are produced for these pages. The volunteers then had the opportunity to review the remaining segmentations. In experiments, the accuracy of BCS with respect to this dataset was found to be slightly lower than that

of VIPS, but the BCS algorithm was considerably faster.

For our purposes, the main interesting points about BCS are its use of a flat segmentation rather than a hierarchical one, and the dataset used for evaluation. Although common in natural image segmentation, a flat segmentation is unusual in segmenting web pages, for which a hierarchical segmentation seems natural. Nonetheless, Zeleny *et al.* argue persuasively that for some applications the leaf nodes are far more important than the internal nodes. While a segmentation tree is preferable for the assistive interfaces we have considered here, since it allows the hierarchical structure to be leveraged in the interface, a flat segmentation may be a valuable tool for other areas. The method by which ground truth segmentations were elicited is interesting as a means of quickly producing ground truth segmentations for a large dataset by leveraging the implementation structure of web pages. It does, however, carry the limitations in expressivity inherent in the use of the DOM tree rather than an image of the page; infographics, for example, would be forced to be leaf nodes in these segmentations. As the method uses blocks produced from DOM tree nodes by the rendering engine, the same limitations apply to BCS (and to VIPS, which was used as a point of comparison), so the ground truth is accurate within the limitations of the algorithms being tested, but the approach would not be suitable for generating ground truth for an image-based segmentation algorithm.

There have been a few prior attempts to use images of rendered pages in web page segmentation. In 2009, Barkol *et al.* proposed a system based on heuristic rules to find pairs of long lines in the image which can be used to define regions, combined with text detection techniques from OCR [12]. This system was intended to be used as a generic GUI parsing system, but the focus in evaluating it was on web pages. Many common elements exist in web pages and program GUIs, especially in forms, and similar design cues are used in both domains. By convolving the image of the page with edge detection kernels for horizontal and vertical lines, a continuous edge map is produced, and from this map line segments are extracted. Using the expected size range of objects in the web page to define thresholds for the lengths of significant lines, pairs of horizontal lines and pairs of vertical lines can be assembled into rectangular regions. The procedure is most effective for objects with a rectangular outline in the edge image. Because text contains many sharp edges, a separate text detector based on techniques from OCR is used to extract the text regions. Foreground and background colour consistency requirements are used to refine the segmentation. This process is repeated hierarchically to produce a segmentation tree. While at the lowest level this is similar to our approach, the edges are assembled heuristically, and used to produce boxes directly rather than to locate divisions. The method proposed by Barkol *et al.* is a heuristic, rule-based approach, as opposed to our Bayesian approach.

To refine the segmentation tree, Barkol *et al.* used a combination of a graph grammar (a generalization of the formal grammars used to define formal languages) and recognition of low-level GUI elements to produce a more semantically-aware classified segmentation tree, which may include additional nodes representing semantic groups. Common design patterns composed of GUI elements are recognized by the roles that they play in the graph grammar. Because, as Barkol *et al.* acknowledge, no grammar can fully capture the structure of all graphical interfaces, this grammatical interpretation of the page structure proceeds from the leaves of the segmentation tree upward until no production rules can be applied. The classification tree is, therefore, not necessarily complete. The grammar used to test the algorithm is relatively small, consisting of eighteen symbols (eight terminal and ten non-terminal) and approximately twenty production rules. Barkol *et al.* presented an example parsing of a specific airline reservation page to demonstrate the capabilities of their segmentation and classification algorithm.

Our work is distinct from that of Barkol *et al.* in a number of important ways. Our segmentation algorithm is more integrated. We use the same Bayesian edge detection method to find the edges of text regions and to detect other significant edges such as changes in background colour. Thanks to our Bayesian approach to adapting to local image characteristics, this method is effective in highly-textured, edge-dense text regions as well as in areas where two empty areas of slightly different background colours meet. The segmentation tree is constructed in the same probabilistic framework, and incorporates a prior probability distribution over segmentation tree structures. Our classification method also uses a very different structure. Our model is based on a probabilistic graphical model, and by inferring the *most likely* joint labelling of regions we avoid entirely the problem of encountering situations not anticipated in the hand-tuned graph grammar. As a result, our parsing system has a much more robust and principled architecture, since it produces a full labelling in each case and all classification rules are learned form the data rather than hand-coded. Additionally, our evaluation methods are far more comprehensive. Rather than a qualitative test on a single page, we show a series of qualitative and quantitative experiments on pages from a range of types, and evaluate the performance of our web page parsing system in a range of contexts. Because our evaluation was more thorough, our work establishes much stronger evidence both for the efficacy of our method and for the efficacy of vision-based parsing of screen images as a strategy.

Cao *et al.* proposed a system based on finding minimal bounding boxes around edges detected in an image of the page. This system was intended to support the identification of "phishing" web pages[2] by supporting comparison between the visual structures of different

[2]Phishing web pages impersonate legitimate sites in order to collect sensitive information from victims.

web pages. The algorithm proposed by Cao *et al.* is based on two fundamental assumptions: first, that web pages are composed of rectangular regions (implicitly, axis-parallel rectangular regions), as we assume in our work; second, that regions in a web page are separated by empty background regions. Once an image of a page has been obtained, it is processed using the Canny edge detector [23] to produce a binary edge map. Cao *et al.* define a sub-image as an axis-parallel rectangular region of an image, represented as a tuple containing an origin in image coordinates, region width and height, and image data in sub-image coordinates. Sub-images can be defined for other sub-images. A "real" sub-image g' is the minimal sub-image of a sub-image g which contains all nonzero pixels in g. The real sub-image is produced from a sub-image by a shrinking process. The "leftover" empty rectangular regions are termed "dividing zones". It is necessary to detect not only a single real sub-image, but a hierachy of sub-images. In order to accomplish this, dividing zones are detected by finding region-crossing lines without edges. This is in some sense the opposite of our approach of finding region-crossing lines *composed of* edges. Divisions are prioritized based on heuristics when multiple divisions are possible. The resulting segmentation trees were compared using a variation on the earth mover's distance to flag possible phishing pages.

In order to establish the efficacy of their page segmentation algorithm as a segmentation algorithm, Cao *et al.* performed qualitative testing on a dataset of "[m]ore than 20 web pages" [24]. It was also tested in the context of phishing page detection, its intended use, and was found to allow more robust detection in this setting.

Unlike our parsing system, the segmentation algorithm described by Cao *et al.* does not perform classification. It also does not use a Bayesian approach to the page segmentation problem, as ours does. That being said, the performance advantage over a competing method in the area of phishing page detection shows the potential utility of the approach in this domain, and the simpler method may provide a performance advantage in offline processing of large numbers of pages, as would be done when attempting to identify all phishing pages in a large quantity of possible spam email. As it is motivated by assistive interfaces, our system must produce a more sophisticated parse tree of the page, but batch processing of very large datasets is less of a concern. Like Cao *et al.*, we use qualitative evaluation of our segmentation algorithm as a segmentation algorithm, and quantitative evaluation in context; due to the differences in the intended application domains, however, these experiments are very different. We also study the characteristics of ground truth segmentations, quantitatively evaluate segmentation quality, and quantitatively compare our segmentations to DOM tree-based segmentations.

More recently, in 2015, Wei *et al.* proposed an interesting combination of VIPS with vision-based segmentation called HoughVIPS [132]. It makes use of a variation on the

Hough transform (commonly used in computer vision) called the progressive probabilistic Hough transform. The classical Hough transform [70] exhaustively searches an image for features, and for each feature found increments an accumulator at each hypothesis that could explain the presence of a feature. The features may, for example, be edge pixels, and the hypotheses may be lines in the image. The progressive probabilistic Hough transform (PPHT) [91] is a variant which prioritizes the examination of features and hypothesis in order to allow the algorithm to be terminated at any time and provide results from the accumulator. In HoughVIPS, PPHT is used on an edge map produced by the Canny edge detector to detect additional separators not found by VIPS. The results were evaluated by four volunteers rating the quality of segmentations; HoughVIPS achieved slightly higher performance than VIPS.

HoughVIPS is an interesting combination of image-based and DOM tree-based techniques. It attempts to address the shortcomings of the DOM tree-based VIPS algorithm by incorporating a simple image-based method. Our method, in contrast, uses a completely vision-based approach, integrating all phases of the image segmentation process in a single probabilistic framework. The approach to evaluation taken by Wei *et al.* is advantageous in that it does not require explicit creation of ground truth segmentations. It is, however, inherently subjective, and is likely to be less sensitive to details than a more standard ground truth-based evaluation method such as ours.

In contrast to these previous attempts to use page images in page segmentation, with our proposed vision based system, we aim to produce a comprehensive framework, supported by thorough evaluation. We also adopt a principled Bayesian approach to the entire parsing problem, including segmentation.

7.4 Web Page Region Classification

Chen, Zhong, and Cook describe a method for using a generalized hidden Markov model (GHMM) for classifying regions in a web page [28]. An ordinary HMM is a probabilistic graphical model in which there is an underlying state and an observation at each discrete time step in an evolving system; the observation and next state depend probabilistically on the current state. In a GHMM, the single observation of an HMM is replaced by a set of observations, allowing a set of features to be used rather than a single feature. The states represent the types of regions in the web page, and the observations represent the relevant features of the regions. The problem of classifying these regions can be viewed as the problem of finding the sequence of states which is most likely, given the observations of visual properties of regions and the dependency of each state on the previous one. A

significant problem that Chen *et al.* needed to overcome in creating a GHMM of web page organization is that a web page has a two-dimensional layout whereas a GHMM assumes a one-dimensional sequence of states. The authors used a depth-first traversal of the tree of regions to create such a sequence of states.

Our system uses a Markovian classification system, but rather than the GHMM of Chen *et al.* it uses a hidden Markov tree (HMT). The hierarchical nature of the HMT directly reflects the hierarchical nature of a web page, unlike the sequence structure used in the GHMM. With a depth-first traversal of the segmentation tree, the successor of each internal node will be one of its children, as expected. The successor of a leaf node, however, cannot be its child, since leaf nodes by definition have no children. As a result, the sequence structure has many transitions which do not properly reflect the hierarchy of the segmentation. Worse, it is likely that the distribution of labels in leaf nodes will be different from the distribution in internal nodes. More formally, for a segmentation tree with n nodes, including n_i internal nodes and n_l leaf nodes, there will be $n_l - 1$ transitions which do not reflect the segmentation tree and n_i which do. Trees in general include a sequence structure with a single leaf and no incorrect transitions, but this is not a plausible model for a segmentation tree; with our X-Y tree prior, it is impossible for any internal node to have fewer than two children, which rules out a linear tree entirely. In the experiments described in Chapter 6, we observed trees with many leaf nodes, such that $n_i \ll n_l$; for these realistic trees, transitions in a sequence model which reflect the tree structure (*i.e.*, transitions from internal nodes) are the exception rather than the rule, while transitions in our HMT model always reflect the segmentation tree structure.

A recent rule-based classification proposed by Akpinar and Yesilada [2] uses the eMine ontology to define both a set of labels and a set of rules. These rules are based on a range of features of regions in the page, including

- HTML tags (including tags introduced in HTML5)
- Parent and child regions in the DOM tree
- Position and size
- Functional attributes such as `onclick`
- Style and visual appearance, such as background colour and font size
- Keyword cues found in surrounding text and in attributes such as `class`

These cues allow evidence from visual features and from the implementation to contribute to the classification of a regions, although these are of course derived from the DOM tree rather than from an image of the page. A large set of sophisticated heuristic rules is used to infer labels for each region. For each region, a likelihood score is tracked for each label; this score is determined by the heuristic rules and region characteristics. The performance of this region classification approach was tested by manually-labelled ground truth produced by participants using a web-based interface. The results were promising, showing an agreement of approximately 71% with participant ratings if equivalent terms were accepted as correct.

Our classification system uses an HMT to infer class labels, rather than a hand-crafted ontology. In one series of experiments (Section 4.4), we used the labels from the eMine ontology to produce manual ground truth labellings for a dataset of segmented pages. Our algorithm achieved approximately 61% accuracy, approaching the performance of the sophisticated heuristics used by Akpinar and Yesilada using only learned information and simpler features; the results we obtained also suggest that higher performance may be achievable by more carefully matching segmentation parameters to the inherent granularity of the ontology. A learning-based system like ours has the additional advantage of easier modification than hand-tuned heuristics.

Kovacevic *et al.* proposed a heuristic method using visual properties obtained from the HTML source code to classify regions as "header", "footer", "right menu", "left menu" and "center", with the objective of improving the classification of entire web pages on the basis of content [79]. By classifying regions, it is possible to focus on the text occurring in central regions that are likely to contain key content, and assign less importance to text in areas such as the footer of the page. This system uses a simplified rendering engine to extract a tree of HTML tags with positions and attributes. The rendering engine does not support CSS; this would have been less of a disadvantage in 2002, when the method was published. The heuristics are based largely on tag type and position; due to the selection of classes, absolute position in the page can be used rather than position relative to a parent region. This method was found to be advantageous in page classification, although the limited selection of classes would correspondingly limit its use in assistive applications.

Ahmadi and Kong proposed [1] a method for detecting major regions of the page based upon position in the page, in order to support browsing on small screens (such as cell phones and other mobile devices). Especially at the time when mobile web browsing was initially becoming popular, web pages often did not support small screens, making automatic adaptation an appealing option; this is similar to the ongoing status of accessibility support on the web. Ahmadi and Kong proposed their classification algorithm as a means of identifying closely-related regions in the page so that these regions can be kept together

while content is rearranged for a smaller screen. This method uses a combination of source code structure and layout structure. The first step is to detect a small set of regions, consisting of those used by Kovacevic *et al.* plus a "clutter" class which includes advertisements. The second step is to divide the main content into sections containing related content. Separate heuristic rules, considering HTML tags, position, and attributes such as background colour and font size, are used in each step.

A method described by Burget and Rudolfova [20] for information retrieval uses a particularly rich set of features in classifying regions. Starting from a segmentation tree produced using visual features of the DOM tree, this algorithm classifies regions using features including average font size (relative to the average font size across the entire page), position relative to its sibling regions, and contrast. In addition to these visual features, five text features are used: text length, number of spaces, numbers of upper-case and lower-case letters, and number of digits. A learned decision-tree classifier is used to classify regions into one of ten classes. Nine of these classes are related to article features such as headings and paragraphs; the remaining class is a catch-all for regions that are not part of the main article.

The earlier heuristic approaches of [1] and [79] share the limitations fundamental to the heuristic approach as described in the context of the eMine ontology. These systems, and a number of closely related ones, also tend to use a small number of classes. This reduces the number of heuristics required and simplifies the process of testing the heuristics, but also limits the expressiveness of the classes. Burget and Rudolfova's approach uses machine learning to perform the classification, rather than hand-tuned heuristics. This alleviates some of the problems with writing heuristics manually, although it does require a substantial labelled dataset. One limitation of the approach described in [20] is that although it uses hierarchy in that the position of each node relative to its siblings provides evidence for its label, parent-child relationships are not used. This may be a result of the classes in the ontology; they pick out different parts of an article at roughly the same level of abstraction. Our approach also uses machine learning, but uses a larger and more general set of classes, and models the complete hierarchy of the page using the HMT in order to require consistency between parent and child labels.

Although not specifically designed for web pages, the "Prefab" GUI analysis system described by Dixon and Fogarty is related to this area [41]. Prefab uses object recognition techniques to identify common GUI elements in a screen image. Barkol *et al.* also used the recognition of GUI elements such as radio buttons to refine their segmentations [12].

7.5 Assistive Technology for the Web

The accessibility research community (including, for example, the Web4All community) has produced a wide range of methods for ensuring accessibility on the web and in "the real world". This research has including accessibility standards, appropriate types of interfaces for different requirements, and usability considerations. In this section we present only a sampling of work in web accessibility that is of particular relevance to our research. Our work is complementary to existing research in assistive technology for the web; rather than presenting a new type of front-end interface, our work provides a new back-end for interpreting content structure in order to provide a better and more robust interpretation of the content to an assistive interface which can then present the content in accordance with current best practices.

Most work to date has focused on improving accessibility for visually impaired users accessing the web via screen readers, although there is much potential for leveraging these same core ideas in other domains. Asakawa and Takagi [11] developed a system that transcodes existing Web pages with manual annotations to add structure that can be used for reordering visually fragmented grouping according to performance. As manually annotating pages is labour intensive, follow-on work focused on automatically annotating similarly structured pages across multi-page websites [121]. Yesilada et al. [136] use a structured ontology to identify visual segments in a web page and re-engineer pages to facilitate navigation using a screen reader. Mahmud, Borodin, and Ramakrishnan [89] introduced CSurf, a system that uses web page partitioning techniques from natural language processing and machine learning to capture the context of a link to enable the screen reader to start reading the next page starting from the most relevant section. HearSay [18] extends this approach to additionally facilitate movement between the segments of a page. Our approach is different in that it is entirely automatic, and focuses on purely visual cues in the rendered page when identifying relevant regions, in order to obtain implementation independence.

Screen reader navigation is often difficult. Appendix D shows an example of a transcript of browsing a complex web page using the NVDA 18.02 screen reader. In this relatively simple task—finding the seven-day forecast in a page from The Weather Network—only a small number of the features of the system were used. Nonetheless, the process is complex, involving navigating by element type as well as moving from one line to the next, and using the default verbosity settings a great deal of information is supplied to the user. It should be noted that this transcript was produced by a sighted novice user of the NVDA program, and as such may not reflect exactly the experience of visually impaired users, but should serve to illustrate the complexity of even apparently simple tasks when using a

modern screen reader.

Some work has been done in using visual properties of web page elements (obtained from the source code of the page) to improve screenreader performance. Krüpl-Sypien *et al.* describe a bottom-up, rule-based system which represents a web page using a layered model, similar to the layered model of a network stack [81]. At the lowest level is the page source code; progressively higher levels contain discrete objects such as words or images, collections of objects, and logical divisions of the page. The top layer provides a model for navigating through the page on multiple "axes" of logical relationships between objects. This model has been applied to screen readers as a part of the Vienna University of Technology's Project ABBA (Advanced Barrier-free Browser Accessibility) [125]. Our proposed approach differs in that it is intended to be independent of the implementation of the web page, using only the rendered web page and not the source code, and to avoid the use of rigid, hand-coded rules in favour of learned characteristics and classifiers.

Some researchers have emphasized the value of addressing clutter in web pages [113, 85]. Some strategies proposed include directing users to make use of buttons in order to bring content located in the core region of the page into focus. Users can also ask for certain parts of the page to be read aloud. With the segmentation algorithms developed in this thesis, it is possible to automatically associate content into visually coherent blocks, which can readily be treated as a unit. Our decluttering method is focused on highlighting a region of interest to reduce the distraction from all other content on the page. Because our approach to decluttering makes use of our interpretation of the visual rendering of the page, this allows for more extensive options for the user. Magnification can also play a role in emphasizing the focus area; since our segmentation allows "smart" magnification that adapts to visually coherent regions in the page, the possibility of increasing distraction by requiring adjustment of the magnification window or moving a fixed window around the area of interest is greatly reduced. Our classification method may also allow content to be suppressed based on its semantic role in the page, further automating the decluttering process. For example, a user might ask for the header and footer to be removed for all pages, and while reading an article may ask for the navigation links in the sidebar to be removed.

Research has also shown that older adults may benefit from personalized solutions, when presenting web pages [83]. Our methods are designed to enable individualized solutions. As shown by the interface described in Chapter 5, users can have a wide range of options for specifying their preferences for interactive zooming and decluttering. In addition, our parse tree of the page represents the visual structure and the high-level semantic structure of the page, and can support complex, high-level modifications of the page by providing a high-quality interpretation of the page structure to a customizable interface. Because our

method is based on Bayesian machine learning, online learning techniques can be applied to allow even the low-level features of the parse tree to adapt to user preferences (*e.g.*, to err on the side of oversegmentation or undersegmentation), and a high-level interface could similarly use online learning to adapt to user preferences. We have begun to explore the user base of older adults (Chapter 5) and have plans for expanding our efforts with this group of participants, as outlined in Section 8.3.1.

Also relevant to the problem of assistive technology for the web is an established field of research on depiction of web pages for mobile devices (*e.g.* [29, 30, 138]). Although improvements to mobile hardware such as smart phones and increasing awareness of the need to design and implement web pages with mobile devices in mind (due to the increasing prevalence of these devices) have reduced the need for client-side page transformation in this area, work in the area of adapting pages to mobile devices is relevant to the challenges we are trying to address in web accessibility. A mobile device with a small screen requires an alternative presentation of the page or an intelligently-selected subset thereof if the user is to browse the page effectively and with minimal frustration. This is related to the problem of presenting a page to a user with limited vision, for whom even a desktop monitor may only be able to show as much content legibly as a mobile screen can for a user with sharper eyesight, or to a user with attention deficits who would struggle to focus on an area of interest if all the content is displayed simultaneously. Exploring the context of mobile web pages is an interesting avenue for future work.

Chapter 8

Contributions and Future Work

The research carried out in the course of this thesis maintained the applied and theoretical tracks described in the introduction. The applied track is focused on the application of computer vision techniques to page parsing for web accessibility, and the theoretical track is focused on the broader implications of our research in computer vision. Our key contributions can be similarly divided into contributions to assistive technology (Section 8.1) and contributions to computer vision (Section 8.2). In this chapter we also discuss plans for future work resulting from the research described in this thesis, in both the theoretical and applied tracks.

8.1 Contributions to Assistive Technology

From the perspective of assistive technology, our key contributions relate to the development of our page parsing system. The system we developed in the course of this research is capable of supporting assistive interfaces by providing a rich parse tree of the semantic structure of a web page. Because it uses only an image of the page to interpret the page structure, it is implementation-independent, and uses exactly the evidence that a human user without assistive needs would use to parse the page.

At a high level, our user study (described in Chapter 5) provides interesting insights into the process of running an experiment to test an assistive interface intended for users with challenges related to specific tasks but who are not fully disabled with respect to those tasks. In our case, our assistive interface was designed to be of use to (among others) users with poor vision who were not blind. We experienced difficulties in recruiting users in this

category, as it was difficult to fully and clearly specify the level of disability we intended to address in our recruiting materials. Although assistive needs are a spectrum, rather than something a person has or does not have, experiments aimed at the middle of the spectrum rather than the extremes have unique challenges, as our experiences show. On the other hand, it was possible to design our magnifying and decluttering interface with a very simple interface because of the abilities of our target audience, thus allowing greater experimental control over user experience factors related to the front-end interface while testing the efficacy of our back-end segmentation algorithm in this domain. When designing an experiment, we have learned that it is worth considering which users in the spectrum of assistive needs should be targeted in order to best achieve the experiment's objectives. In addition, it is important not to neglect users in the middle of the spectrum of assistive needs when designing assistive interfaces, since the appropriate assistive technology is not necessarily the same as for users at the extremes of the spectrum.

Our use of offline testing in evaluating the interface described in Chapter 5 is also relevant to the process of conducting user studies. By applying an evaluation method based on simulated input, we were able to evaluate the degree to which the combined system performs as designed. This experiment was far simpler than a complex user study. The aim was to offer a replacement for real user input, with a view that provided the simulated input generates a superset of true patterns of usage, this approach will be effective. Based on our experiences, we recommend the use of offline testing as a component of an experimental evaluation strategy for assistive technology, since it allows thorough testing of key functionality and the theoretical capabilities of a proposed interface before experiments with users are conducted to establish hands-on usability.

Our segmentation algorithm provides a high-quality hierarchical segmentation of a web page into visually and semantically coherent regions. This segmentation can be used directly to support a number of assistive interfaces, depending on the specific requirements of the users. A blind user, for example, would require an audio presentation of the page contents, while a user with some vision might prefer an interface based on magnification or increased contrast. As demonstrated in Section 3.4 and Chapter 5, intelligent, content-sensitive magnification and highlighting can both be accomplished with a segmentation tree of sufficient quality. Using a segmentation tree allows the area of interest to adapt to the page, rather than using a fixed-size window (which may be either too small or too large for the true area of interest) or a manually-adjusted window (which adds to users' cognitive burden and slows down browsing). For these reasons, our segmentation algorithm represents an important advance in web page segmentation.

Our hidden Markov tree (HMT)-based classification algorithm also represents a significant contribution. Although Chen *et al.* [28] used a related generalized hidden Markov

model, the sequence structure of this model does not accurately reflect the inherent tree structure of web pages. By allowing the structure of the HMT to adapt to the structure of the segmentation tree, our classification algorithm allows the structure of the probabilistic graphical model used to perform inference to match the inherent structure of the domain. Our results (in Chapter 4) demonstrate the promise of this method despite the challenging nature of the problem of classifying regions in a web page according to high-level semantic roles.

As it is based on a Bayesian framework, our page parsing system can be readily extended to incorporate forms of evidence other than the visual structure of the page, in the common framework of conditional probabilities. Unlike a heuristic-based system such as VIPS [22], it can be readily extended to use both evidence from the DOM tree, including assistive features where they are available, and visual evidence from the rendered page. This represents a substantial advantage, since a complete assistive interface designed for use in practice should use all of the available information to provide the best possible performance for users. In the research presented here, we have used vision exclusively, since new developments in vision-based page parsing are the main focus of our work.

Compared to the few other proposed systems that make use of computer vision techniques in understanding web page structure, our page parsing system represents a significant advance. It is more complete, incorporating both segmentation and classification. It is more principled, using a Bayesian approach constrained by fundamental features of the domain rather than heuristic rules. It is also more comprehensively tested, with extensive quantitative and qualitative tests of a wide range of aspects of the system's performance. These tests include individual tests of the segmentation and classification components of the system in isolation, as well as tests of an assistive interface based upon the segmentation tree produced by our system. This system, therefore, represents a new standard for the development of these systems with promise for practical use.

By supporting improved assistive technology for the web, we hope to make using the web easier and more enjoyable for users with assistive needs. As the web becomes more and more integrated into daily life and as the population ages, the problem of web accessibility is becoming increasingly important. At the same time, although standards, design guidelines, and technologies exist for creating accessible web pages, accessible web page development requires additional effort and expertise; as a result, these frameworks are not consistently and reliably used. Client-side frameworks that can interpret web pages that are not designed with accessibility do not rely upon accessibility support being built into the design or implementation of web pages, and therefore avoid this problem. Our vision-based parsing system is designed to support such client-side assistive technology by supplying sophisticated, implementation-independent analysis of the semantic structure of the page to

an assistive interface, which can then modify the presentation of the page according to a user's needs. Our work provides an innovative back-end page parsing system, with a focus on ensuring that this system is functional, in order to then provide a better basis for the front-end interfaces designed by HCI researchers to present the content of a page to users. Our work is therefore complementary to the extensive work done on the proper methods for adapting interfaces for users with assistive needs. While we do not claim to solve the problem of web accessibility, our research represents a considerable advance in the area of vision-based page parsing, and contributes to the overall goal of a web accessible to all. We note that our architecture, with its Bayesian approach, encourages interpretability and makes the use of intermediate results (such as the unclassified segmentation tree) simple and convenient. Interpretability may be especially advantageous when working with the developers of complementary front-end interfaces.

8.2 Contributions to Computer Vision

Although our initial motivation for research is in the application area of assistive technology, our primary scientific objectives are in the field of computer vision. At a high level, we have contributed to the field of computer vision by providing a systematic investigation of rendered web pages as a dataset for computer vision research and as a distinct class of images. We have labelled our study as one of man-made images and return in Section 8.3.3 to discuss some extensions to this investigation. Our experiments show that images of web pages are a rich class of images for research. Many useful assumptions can be made: due to the rendering process, perspective effects and blurring do not occur; regions in the segmentation tree can be assumed to be axis-parallel rectangles; the segmentation tree can be approximated by an X-Y tree. On the other hand, there are important challenges in parsing these images. High-level cues such as long-range alignments are important in inferring the structure of the page. Major cues such as edges are highly context-dependent; edge strength, for example, is a poor guide to the significance of an edge due to the use of slight changes in background colour and faint dividing lines to convey high-level structure in the page. The combination of useful structural assumptions and subtle cues to the true structure of the page make these images an appealing domain for vision research.

Our research into the elicitation of ground truth segmentations, described in Chapter 6, provides insights in two key ways. In the specific area of perception of web page organization, our results help to determine the number of participants required to segment each page by showing the degree of difference between users; as a result, "mass production" ground truth generation can be performed efficiently. The results from this study

also demonstrate the existence of a performance ceiling on any "one-size-fits-all" segmentation algorithm, which is useful for the evaluation of segmentation algorithms and for the determining whether such a one-size-fits-all method will be satisfactory to all users. More broadly, this study provides valuable insights into methods for eliciting ground truth segmentations. In the process of designing our experiments, we were forced to consider the best approach to segmentation elicitation given constrained resources and a specialized domain. Our approach of performing a study of a small number of instances, each segmented many times, to assess the properties of segmentations and the degree of between-user differences prior to large-scale work proved successful, and should be applicable to improving the efficiency of segmentation elicitation in other domains. Efficient elicitation of ground truth segmentations is important due to the high cost of ground truth data for segmentation algorithms, and is especially relevant for specialized domains in which large-scale projects are less practical than in more common domains such as natural scenes or medical images due to lack of resources. Our study also raises the question of which starting point is best when eliciting segmentations. The "empty" segmentation provided when users "start from scratch" is, after all, a segmentation itself, consisting of a single region encompassing the entire image; it is entirely possible that a different segmentation, perhaps produced by a segmentation algorithm, would be a more appropriate starting point for users to produce ground truth segmentations from. The answer to this question is likely to depend upon the domain of images and perhaps the intended use of the ground truth segmentations.

Our classification algorithm represents a novel model for the classification of regions in a web page. Using a hidden Markov tree (HMT) with a structure corresponding to the structure of the segmentation tree, in which hidden states represent object classes, we have achieved good performance in a very challenging classification problem. In our classification method, the structure of the segmentation tree is represented not by structural features in the feature vectors describing regions, but in the structure of the graphical model used to infer the most probable assignment of labels. By requiring that all nodes share the same probability distributions, a single set of distributions can be learned from segmentations of varying structures and applied to inference for segmentations of varying structures. Although designed for the problem of classifying regions of a web page, our classification algorithm could be readily adapted to other domains by using alternative features (such as shape and texture) suited to the domain. The classification algorithm requires only a segmentation tree and a set of features defined on regions in the tree; this is obtainable for many domains, not just for web pages. As a result, this algorithm represents a key contribution to computer vision as a field.

Our parsing system as a whole is an explicitly Bayesian system for segmentation and classification. Furthermore, the estimated probabilities are normalized, and treating these

values as genuine probability values when, for example, setting thresholds, produces good results. This strongly suggests that these estimates are reasonably accurate. Our parsing system is therefore well-suited to performing computer vision experiments with explicitly probabilistic methods in a restricted but nontrivial domain. Studying the effects of different prior probability distributions over segmentations, for example, is quite feasible using our system, while in other domains it is extremely difficult to specify any but the simplest distributions. As discussed in Section 2.5 and Appendix C, Bayesian methods are used in computer vision; these methods, however, often produce unnormalized probabilities, which act as an energy function consistent with probabilities but cannot be directly used as probabilities in setting thresholds and are more difficult to combine across domains.

8.3 Further Research

The research described in this thesis has raised further questions to be examined in future work. Our plans for future work include follow-up experiments based on our user studies, further development of our page parsing system, and further research into the connections between our work and other areas of computer vision.

8.3.1 Follow-up Studies

Using the data obtained from Chapter 6's experiments in segmentation elicitation for web pages, we would like to gather a much larger dataset suitable for broader dissemination as a standard dataset of web page segmentations. This would be valuable for evaluating web page segmentation algorithms in general—including DOM tree-based methods as well as vision-based methods—and for studying segmentation characteristics in detail with a larger dataset. Crowdsourcing, through platforms such as Amazon Mechanical Turk (AMT), is one possible method for obtaining large numbers of ground truth segmentations. By leveraging large numbers of participants, each performing a single segmentation, a large number of individual segmentations can be readily obtained; since the participants work remotely, carrying out segmentations online, the logistical requirements are dramatically reduced compared to a local method in which participants come to a central location supervised by researchers. In this model, the participants may only complete a single segmentation each; because we performed the study described in this thesis using a small set of images segmented by larger numbers of images, we have less need for multiple segmentations from each user. The crowdsourcing approach does require a segmentation tool which can be run in a web browser. During the implementation process of the segmentation system

used here, we considered the possibility of using AMT or a similar service, and accordingly implemented the front-end interface in Javascript. This will allow rapid conversion from a system running on a local machine for sessions held in a lab to a system running on a central server for users to access from their own systems. We performed our experiments in person in order to ensure that the segmentation interface is usable (and provide further instructions if participants encountered any difficulties), and to obtain multiple segmentations from each user for comparison between users. With a crowdsourcing method, participants may only complete one example, take their payment, and move on to another task. Having verified the usability of our interface and obtained a dataset with multiple segmentations for each participant, it is now appropriate to use crowdsourcing to obtain a larger-scale dataset. Researchers including Rosenfeld *et al.* [112] in fact point out the importance of using a combination of in-lab and AMT tests.

It would also be valuable to deepen our understanding of the results of the user study that we already conducted and described in Chapter 6. We are interested here in pursuing further study of the similarities between participants' ground truth segmentations by applying more sophisticated clustering techniques to identify structure in these relationships.

The user study described in Chapter 5 is another area where expanded follow-up experiments are desirable. In particular, using a larger group of users, screened more carefully to eliminate users who do not need the assistive technology being tested, should provide stronger results about the performance of our back-end parsing system in a practical assistive interface. For those users employing the assistive interface, it may also be valuable to expand data collected to also track gaze, though a careful choice of equipment would be needed, especially if working with older adult participants.

Another possible approach to the follow-up user study would be to test a screen reader, rather than a magnifying-and-decluttering interface; this would allow more stringent and specific criteria for participant selection and would reduce or eliminate the problem of users accomplishing the task without using assistive features. Testing a screen reader as the assistive interface rather than a magnifying and decluttering interface would require a much more complex testing system. The back-end parsing system would require classification as well as segmentation, so the aim of the experiment would be to establish the utility of the entire segmentation-classification pipeline in the context of assistive technology. The interface itself would also be much more complex, and would require a sophisticated navigation model that would not be confusing for users with experience using commonly deployed screen reader programs such as NVDA. Although these challenges are significant, they are not insurmountable, and such an experiment would be of considerable value in exploring how to best use visual parsing in assistive technology.

We are especially interested in exploring further how to improve the experiences of users with online social media, such as Facebook. The pages displayed here are at times excessively cluttered, posing challenges for certain users, including older adults. Research has in fact shown that senior citizens are increasingly invested in becoming Facebook users [73], so that it would be valuable to determine whether decluttering in this particular context is especially beneficial.

In general, continuing to study older adults as a possible user base for our methods should be of value. As indicated by Dickinson [40], when these users struggle with computer use, they may simply abandon the attempt and our aim would be offer improved online experiences. A variety of efforts to improve the interface experiences of older adults, from using digital pen and paper email systems [104], to enabling better connections to family [48, 93], provide some evidence that participants from this community will still be interested in learning of novel opportunities to engage with computer technology. To date, we have benefitted from collaboration with HCI researchers (including those with expertise in assistive technology). For the future directions outlined here, we would plan to continue such collaborations.

With respect to offline testing, learning actual user behaviour by studying a set of real users may help to suggest improvements to our simulations. For example, the noise that is added in order to simulate user behaviour could potentially be modelled on actual deviations observed. Doing so would assist in drawing out the practical benefit of the functions provided in the interface.

8.3.2 Algorithm Improvements

We are very interested in extending and improving our proposed segmentation and classification algorithms. The results obtained in our offline tests of the prototype interface (see Chapter 5), combined with qualitative examination of the results, suggest that the segmentation algorithm succeeds very well in many regions, but fails in other (*i.e.*, the segmentation errors are not evenly distributed across regions). Often, these errors result in segmentations with an implausible structure. We may be able to improve performance by using a more sophisticated prior probability distribution over web page structures. The current prior is uniformly nonzero for X-Y trees and uniformly zero for all other segmentations; a more sophisticated version would allocate higher probability to "web page-like" X-Y trees than to segmentation trees that have a less plausible structure (see Figure 8.1). One approach to improving the prior probability distribution would be to use the ground truth segmentations produced by the experiments described in Chapter 6 to learn the distributions of structural features in ground truth segmentations. The work already described

(a) Plausible (b) Implausible

Figure 8.1: Plausible and implausible X-Y trees for web page structures. With the current prior probability distribution over web page segmentation trees, the two are considered equally likely; a more sophisticated prior could allow the plausible structure on the left to be assigned a higher probability than the implausible structure on the right.

in Chapter 6 on analysing these distributions to learn about users' segmentations also represents a first step toward using ground truth segmentations to improve the structural prior.

Another area of interest for future work is improving our model of individual lines. Lowe, for example, researched perceptual grouping, developing Bayesian methods for determining whether features in the image occur accidentally or are significant in the real scene [88]. Lowe's methods allow tentative conclusions about small structures to be reinforced by observations about the embedding of small structures in larger ones. Although Lowe was focused on images of three-dimensional scenes, accidental alignments can also occur in a web page. Systematic grouping of areas where edges are highly probable could be a useful intermediate stage between calculating the probability of an edge at an individual pixel and the construction of region boundaries, especially for bridging gaps in an alignment edge. Related work by Jepson and Mann analysed similar types of features in terms of qualitative probability [72]. Other possible modifications to the line model may include applying the minimum proportion of lines recursively (so that larger gaps are permitted for larger scale lines). This would require attention to ensuring that the lines are not allowed to continue across other features, but may be useful in ensuring that gaps can be bridged given sufficient evidence of long-range alignment.

At the lowest level, it may be worth exploring the possibility of relaxing the assumption of independence between the edge detector responses at pixels within a given neighbour-

hood. A more sophisticated model, such as a conditional random field (CRF) or Markov random field (MRF) model, would allow the system to account for statistical dependence between nearby edge detector responses, because these models incorporate explicit models of dependencies between pixels in a region, resulting in a probability model which captures the true local structure of real web pages more accurately.

Further research in classification is also of great interest. Our results in Chapter 4 indicate that our classification method is capable of achieving impressive performance despite the difficulties inherent in the problem of classifying regions in a web page, but further improvements to its accuracy would substantially increase its practical utility. We are interested in pursuing further research into improving the training dataset, both by expanding it and by using segmentation trees better matched in terms of granularity to the ontology (*i.e.*, using stopping or pruning conditions that result in leaf regions comparable in size to the leaf regions of the ontology). This research would help to establish the upper bounds for the performance of our HMT-based approach. We are also interested in applying the classification method in other domains, including natural images, to further explore its capabilities. Depending upon the results of these experiments, it may be worth experimenting with variations on the HMT structure, such as using a tree-structured but undirected probabilistic graphical model.

Another interesting area for future research is the generalization of our method to dynamic pages. Moving and dynamically loaded content are common in modern web pages, and present challenges for segmentation and classification. These features also present opportunities. Common movement or appearance properties of regions could be a valuable grouping cue. Similarly, the dynamic properties of regions could be useful in classification; menus, for example, often appear in response to user interaction. Extending our work to this domain would provide an opportunity to work with a new set of visual features and cues, suggesting extensions to our current methods.

It may also be interesting to expand our solutions to consider new efforts to augment the representation of web pages for accessibility. One interesting approach to web accessibility is crowdsourcing the process of providing annotations or transcoding web pages. The WebInSight framework proposed by Bigham *et al.*[16] uses several methods to provide alternative text representations of images, including accessing a manual labelling service as a fallback option. The two primary techniques in this area are applying OCR to an image with text, and using title and heading information from the page linked to by the image (assuming that the image is a link). The AccessMonkey framework proposed by Bigham and Ladner [15] allows users and web developers to collaboratively produce a library of Javascript scripts for performing transcoding operations. Bigham and Ladner provided, among other scripts, an implementation of the WebInSight algorithm. The aim

of the authors was to address current shortcomings with web page designs, where sufficient attention to accessibility may be absent. With our framework, we aim to provide assistance to users with accessibility needs, regardless of the effort made by web designers. We are interested in pursuing possible methods for leveraging additional information provided by web designers or, as in the case of the work of Bigham *et al.*, by other users in a crowdsourcing framework, to improve the performance of our parsing algorithms.

Efficiency is also an area of potential improvement. It is worth noting that our segmentation algorithms were implemented as instrumented testbed programs, and used in batch processing rather than real-time applications. As such, they have not been optimized beyond the level of performance required for convenient offline testing, and require a considerable time to perform a complete segmentation of a page. For many applications, optimization would be desirable to achieve real-time performance. One area with the potential to improve performance is the calculation of the probabilities that line segments are semantically significant. The Monte Carlo simulation, which allows us to avoid combinatorial explosion from exhaustively considering all combinations, is still computationally expensive, but could, in many cases, be cut short after it becomes clear that the probability of a semantically significant line is very low. This could substantially reduce the cost of this step, and similar measures could be applied to other cases of very low probabilities. As is the case with our segmentation method, our classification method has not been optimized for real-time use. Fortunately, belief propagation is relatively efficient for a tree-structured model such as our HMT. Possible areas for future work on optimization include limiting the number of samples used in kernel density estimation and conventional code optimization.

Exploring how to supplement the visual evidence of the web page with evidence from the implementation of the page is also worth considering as an avenue for future work. The research described in this thesis has been focused on studying how to build a web page parsing system for this class of visual image, with complete implementation independence. If we move on to examine the practical applicability of the system, we would be interested in seeing where performance can improve if implementation evidence can, for instance, clarify cases where the visual evidence is ambiguous.

It might also be instructive to make use of big data in order to fine tune our algorithms. One interesting avenue for future research is to use convolutional neural network (CNN) models to extract the data from massive sets of current web pages (covering a wide variety of types of web pages). This kind of processing may be helpful in recognizing all locally significant edges or in estimating the probability that lines are semantically significant by leveraging the ability of these models to learn sophisticated representations of image features. Extensive crowdsourcing could also be employed in order to recover ARIA labels that have been excluded from some web page implementations.

A final direction for expanding our algorithms would be examining possible adjustments if electing to operate with higher-level classes for the region labelling, ones that are sensitive to the meaning of the actual textual content of the page. Selecting these classes should ideally make use of extensive studies of the target user group to determine what high-level labels are most relevant. This is yet another way in which greater engagement of our user community may become valuable.

8.3.3 Extension to Other Applications and Images

One possible thread for future research is providing greater insights to web page designers. For example, the parse tree produced by our web page parsing system could be used by designers as a tool in adding assistive features to the implementation and ensuring that these features align with their intended visual appearance. Combined with realistic prior probability distributions over web page structures (discussed as a direction for future research in Section 8.3.4), our pipeline could also assist in determining quantitatively how closely a given design resembles other pages on the web, in order to assess objectively how "surprising" the design can be expected to be for users.

Our work thus far has been primarily focused on web pages as viewed on desktops. Mobile pages, although related, must be designed for legibility on a smaller screen. Studying these pages could reveal more information about the cues used in web pages of various types, and the importance of mobile browsing makes this an attractive area of research. A comparison between the appearance of the same page in its mobile and desktop versions could provide interesting insights into the similarities and differences in these domains. This comparison could also provide valuable evidence for semantic grouping in the page: objects that remain connected or in close proximity in both versions are more likely to be related than regions which move separately. We suspect that many of the assumptions that we make about web page design would be especially relevant when these pages are depicted on mobile devices; in this respect, this context for studying our models may be particularly valuable.

Another area of interest is generalizing the methods developed for web pages to other types of screen images and complex documents. Windowing systems and program GUIs are especially interesting in this context. The prior over segmentation structures would need to be altered due to the ability of windows to overlap and occlude each other. Additional cues would be available for segmentation, however, including recognition of GUI elements such as buttons and window borders. This suggests that low-level object recognition should be carried out simultaneously with segmentation. The techniques developed for the Pre-

fab GUI analysis system [41] would be applicable for this class of images. Higher-level recognition may still be better performed after segmentation as a separate step.

Within a single application, a GUI parsing system could allow assistive interfaces to work with a program in the absence of API support provided by the designers. Similarly, it could be used to support task automation in applications with no scripting language support. A system capable of parsing screen images in general would open new possibilities for assistive interfaces. Conceivably, a stand-alone assistive "monitor" could be developed to plug into a standard monitor port, parse the screen data sent by the main computer, and output an alternative presentation of the screen suitable for the user. Obviously such a system would have significant limitations in the absence of a fully general solution to computer vision; describing natural images, for example, is an entirely separate problem, although caption generation is an active area of research and has produced promising results (see, *e.g.*, [134]). Even with limitations, however, a stand-alone screen reader device that can plug in to any system without requiring local software, and make basic applications accessible without API support, would be very valuable for visually impaired users.

Generalizing our work to other image classes may require other extensions to our image model. Although ridge and step edges are sufficient to model typical semantically significant edges in web pages, in other image classes it may be necessary to include roof edges. These occur in natural images, and may occur in other man-made images as well.

Moving away from screen images, there are many other interesting types of man-made images that may be worth studying. By studying image and text organization from a variety of cultures and time periods, in art and documents intended for a variety of purposes, such as Egyptian papyri and early modern printed material, we may be able to identify common features that are fundamental to visual organization, and features that are culturally determined, from the perspective of computer vision rather than from the perspective of art history. In such a study it may be necessary to isolate the man-made visual features from the natural visual features of the substrate (*e.g.*, removing papyrus fibres from a scanned image of an Egyptian heiroglyphic text, or removing foxing and stains from a scan of a seventeenth-century broadsheet) in order to study the man-made features alone. Deep learning and convolutional neural networks may prove valuable here for analysing these naturally-occurring textures due to their proven ability to date to learn domain-specific features.

8.3.4 Image Statistics

As for more "pure" computer vision research, we are interested in a more detailed study of the statistical properties of images of web pages, and a comparison with the known statistical properties of natural images. Intuitively, it is reasonable to expect that a randomly generated array of gray levels will not look like a natural image or like a web page. In other words, $n \times m$ natural images and web page images are a small subsets of $\mathbb{R}^{mn}$ [54]. We are interested in characterizing the statistical properties of images falling into these classes; in particular we are interested in characterizing web page images, since the statistics of natural images have been well studied. This research will provide more specific insights into the similarities between natural images and man-made images despite their different appearances.

One of the most important statistical properties of natural image is scale invariance: natural images show similar statistics at a range of scales. This underlies the usefulness of computer vision techniques such as the image pyramid (in which the image is repeatedly downsampled to produce representations where larger and larger features are represented by the same number of pixels) [94]. If an image is scale invariant, it is reasonable to apply the same algorithm at multiple scales; if not, it is likely to be more appropriate to adapt the algorithm separately for each scale. In practice, scale invariance holds approximately for natural images [4]. Whether the assumption of scale invariance is appropriate for rendered web pages is an open question. This would affect both the design of purpose-built algorithms for web pages and the process of transferring algorithms designed for natural images to the domain of web pages.

Another area of image statistics is identifying components which can be linearly combined to produce images in a given class, or identifying filters that respond to these components. There are a range of techniques which can be used in this context. Principal components analysis (PCA) is perhaps the simplest technique, although it tends to produce many simple components in the form of gratings or checkerboards on datasets of natural images. Other approaches to identifying components produce other results. Applying independent components analysis (ICA) to natural images can produce localized, oriented filters similar to receptive fields in mammalian visual systems [98, 13]. Identifying the fundamental components of images of web pages would be useful for several reasons. The properties of these components could indicate similarities (and differences) between natural images and web pages at the level of receptive fields in the visual system, and could therefore provide insight into human perception. These components could also be used to produce high-quality, low-dimensional features to represent the appearance of a region in the web page in order to improve the accuracy of our classification process by using more

informative features.

We are also interested in studying models of plausible web page structures, and what arrangements of regions in a segmentation tree "look like" web pages. Currently, our most advanced segmentation algorithm (Section 3.3) uses an X-Y tree prior to represent the structure of web page images. This is largely consistent with web page structures, although it does not include overlays, and represents an advance over the *ad hoc* prior used in our initial segmentation algorithm (Section 3.2). It is, however, a very general prior, and does not differentiate between more-plausible and less-plausible X-Y tree structures. A more comprehensive and detailed model has the potential to improve the performance of the segmentation algorithm by eliminating implausible structures from consideration. Our studies of classification of web page regions indicate that a detailed model of region label hierarchy is useful in the classification process (see Section 4.4); we are very interested in studying the effect of similar modelling at the level of segmentations. Extended exploration of image statistics would also be helpful in quantifying the prevalence of the various assumptions made about web page design used in constructing our Bayesian models for segmentation.

8.3.5 Personalization in Computer Vision

Finally, we would like to examine in greater depth the possibility of individualized computer vision solutions. Our examination of the differences between the ground truth segmentations produced by different users in Chapter 6 indicates that there is a ceiling on the performance of a "one-size-fits-all" segmentation algorithm. Personalization could allow an algorithm to produce results that more closely match each individual user's perception of the organization of the page. Similarly, user preferences about error characteristics may vary. One user may prefer oversegmentation to undersegmentation, while another has the opposite preference; for classification, different users may put a higher priority on recall rather than precision for specific classes, based on their own browsing style.

Personalization could also play a role in classification. Even using a standardized ontology across all users, it is likely that there will be some borderline examples which could plausibly be considered examples of more than one of the available classes, whether due to a segmentation error, an unusual website design, or inherent ambiguity in the ontology itself. By studying users' perceptions of appropriate labels, it will be possible to determine how commonly different users assign different labels to such regions. A parsing system that uses reinforcement learning to adapt to users' preferences in the handling of these edge cases would allow the system to consistently provide results that fit each user's expectations, rather than requiring users to adapt to the way the algorithm handles edge

cases. For example, when distinguishing complementary content from the main content, one user may prefer the system to err on the side of classifying a region as complementary, while another may prefer that the system err on the side of classifying a region as main content (perhaps because he or she usually skips complementary content and does not want to miss anything important). If users disagree frequently, the ability to adapt to individual preferences has the potential to significantly improve user experiences with systems that rely upon the page parsing system as a back-end. This may also enable a valuable connection to the AI subfield of user modelling [77].

Because our parsing system is intended to support interactive user interfaces rather than autonomous action (as would be the case in, for example, a robot vision system), user preferences play an important role in the perceived performance of the system and therefore in the practical performance of the system. Personalization has great potential to improve the practical utility of our system. Research in this area would also align with the current interest in human-in-the-loop AI systems [67], and could provide insights into how to best adapt computer vision techniques to these systems.

8.4 Conclusion

This thesis has presented an approach for using computer vision to understand web pages. There are two primary contributions: providing improved experiences for the users of web pages (including users with assistive needs), and offering insights for computer vision researchers for understanding semantic cues in images. We have labelled our work as a study of man-made images and reflected briefly on how the solutions developed here may be of use for contexts other than web pages.

Various design decisions have led to the successful outcomes of the thesis. The domain of web pages has allowed examination of a principled analysis of properties of a class of images which goes beyond toy problems and has the potential to inform the study of natural images as well. The Bayesian approach adopted for the modelling enables the segmentations performed to be based on plausible probabilistic reasoning. Our methods encompass edge detection, determining locally significant edges, progressing to semantically significant lines, producing the underlying segmentation tree. The segmentation solution feeds forward to a classification algorithm in a pipelined fashion, enabling various ultimate back-end applications, such as supporting assistive technology.

Novel elements integrated into the models we developed, as we progressed to the final framework included: *(i)* the use of the earth mover's distance in order to examine validation

of our methods; *(ii)* the introduction of the X-Y tree into the segmentation process in order to refine the segmentation process and to leverage a reasonable set of assumptions about this restricted class of image; *(iii)* the integration of an HMT model to support the reasoning for classification; and *(iv)* inclusion of various user studies in order to provide greater insights into the circumstances under which our model is most effective in delivering benefit to users.

We acknowledge that rendered web pages, as a class of images, have the advantage of supporting various limiting assumptions. One of the most important assumptions is that the semantically significant regions in the page are axis-parallel rectangles. This dramatically reduces the search space for segmentations. We also assume that regions are supported by edges, including both continuous and broken step and ridge edges, although many edges occur that do not indicate a region division. Other useful assumptions include the absence of perspective and blurring effects, and that regions do not overlap without containment relationships. The last of these is not invariably satisfied in all pages, but has been found to be a useful assumption in practice.

There was a valuable progression from our first segmentation algorithm, described in Section 3.2, to the one we ultimately settled on using, described in Section 3.3 (and leveraged in our two user studies). These improvements included multiscale edge detection, an improved recursive model of the probability of semantically significant lines, and an improved X-Y tree prior over segmentation trees. The primary practical benefit derived from this revised segmentation solution was improved performance and the ability to address several common segmentation errors such as "overshooting" true boundaries. There was a useful progression as well between our first attempt to validate our classification algorithm and the decisions we ultimately made with respect to the ontology and the source of ground truth data, although we found that the same features and algorithm performed well given proper training and evaluation data.

The various methods used as part of our validation efforts may shed light on how to make this process manageable for any computer vision project, including the value of pruning, the potential for personalization and some insights into how to establish ground truth, especially in the difficult area of obtaining ground truth segmentations. Our dataset of ground truth segmentations (Chapter 6) is also a valuable contribution, both for evaluating segmentation algorithms on web pages and for studying the characteristics of ground truth segmentations in this domain.

We also see noticeable advances simply for the study of web pages, as outlined in several of our discussions in this thesis: the relative advantages of not being tied to the web page implementation, and potential improved opportunities for users with visual or cognitive

deficits.

In all, we present a parsing system for interpreting web page structure based purely on an image of the page, with corresponding advances in computer vision. Our work relates to and ties together many areas, including image segmentation, classification, and assistive interfaces, and requires innovative solutions to problems in each as a result of the characteristics of the domain. Per the discussion in Chapter 7, our methods align well with established strategies adopted by other researchers in the past, but integrate clearly original and distinct approaches. Our work represents an important advance in an application area with significant social implications, and contributes to the understanding of image parsing in unconventional domains.

References

[1] Hamed Ahmadi and Jun Kong. Efficient web browsing on small screens. In *Proceedings of the working conference on Advanced visual interfaces*, pages 23–30. ACM, 2008.

[2] M. E. Akpinar and Y. Yesilada. Vision based page segmentation algorithm: Extended and perceived success. In *Lecture Notes in Computer Science: Current Trends in Web Engineering*, pages 238–252. Springer International Publishing, 2013.

[3] M Elgin Akpinar and Yeliz Yeşilada. Discovering visual elements of web pages and their roles: Users perception. *Interacting with Computers*, 29(6):845–867, 2017.

[4] Luis Alvarez, Yann Gousseau, and Jean-Michel Morel. The size of objects in natural images. *Advances in Imaging and Electron Physics*, 111:167–242, 04 1999.

[5] Aaron Andersen and Cyndi Rowland. Improving the outcomes of students with cognitive and learning disabilities: Phase I development for a web accessibility tool. In *Proc. 9th Int. ACM SIGACCESS Conf. on Computers and Accessibility*, Assets '07, pages 221–222, New York, NY, USA, 2007. ACM.

[6] Apostolos Antonacopoulos. Page segmentation using the description of the background. *Computer Vision and Image Understanding*, 70(3):350 – 369, 1998.

[7] Apple. Apple - accessibility - OS x - VoiceOver. http://www.apple.com/accessibility/osx/voiceover/. Accessed 2014-03-16.

[8] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2011.

[9] Andrew Arch. Web accessibility for older users: Successes and opportunities (keynote). In *Proceedings of the 2009 International Cross-Disciplinary Conference on Web Accessibililty (W4A)*, W4A '09, pages 1–6, New York, NY, USA, 2009. ACM.

[10] Anurag Arnab and Philip HS Torr. Bottom-up instance segmentation using deep higher-order crfs. *arXiv preprint arXiv:1609.02583*, 2016.

[11] C. Asakawa and H. Takagi. Annotation-based transcoding for nonvisual web access. In *ASSETS 2000*, pages 172–179. ACM Press, 2000.

[12] Omer Barkol, Ruth Bergman, Ayelet Pnueli, and Sagi Schein. Semantic automation from screen capture. Technical Report HPL-2009-161, HP Laboratories, 2009.

[13] Anthony J. Bell and Terrence J. Sejnowski. The "independent components" of natural scenes are edge filters. *Vision Research*, 37(23):3327–3338, 1997.

[14] Jennifer C. Romano Bergstrom, Erica L. Olmsted-Hawala, and Matt E. Jans. Age-related differences in eye tracking and usability performance: Website usability for older adults. *International Journal of HumanComputer Interaction*, 29(8), 2013.

[15] Jeffrey P. Bigham and Richard E. Ladner. Accessmonkey: A collaborative scripting framework for web users and developers. In *Proceedings of the 2007 International Cross-disciplinary Conference on Web Accessibility (W4A)*, W4A '07, pages 25–34, New York, NY, USA, 2007. ACM.

[16] J.P. Bigham, R. S. Kaminsky, R. E. Ladner, O. M. Danielsson, and G. L. Hempton. Webinsight: Making web images accessible. In *Proc. 8th Int. ACM SIGACCESS Conf. on Computers and Accessibility (ASSETS 06)*, 2006.

[17] Mark Birbeck, Richard Schwerdtfeger, T.V. Raman, Steven Pemberton, and Shane McCarron. XHTML role attribute module. W3C note, W3C, December 2010. http://www.w3.org/TR/2010/NOTE-xhtml-role-20101216/.

[18] Y. Borodin, J. Mahmud, I.V. Ramakrishnan, and A. Stent. The hearsay non-visual web browser. In *Proceedings of the 2007 International Cross-disciplinary Conference on Web Accessibility (W4A 2007)*, pages 128–129, New York, 2007. ACM.

[19] Yevgen Borodin, Jeffrey P. Bigham, Glenn Dausch, and I. V. Ramakrishnan. More than meets the eye: A survey of screen-reader browsing strategies. In *Proc. of the 2010 Intl. Cross Disciplinary Conf. on Web Accessibility (W4A)*, pages 13:1–13:10, New York, NY, USA, 2010. ACM.

[20] R. Burget and I. Rudolfova. Web page element classification based on visual features. In *Intelligent Information and Database Systems, 2009. ACIIDS 2009. First Asian Conference on*, pages 67–72, April 2009.

[21] Radek Burget. Automatic document structure detection for data integration. In Witold Abramowicz, editor, *Business Information Systems*, pages 391–397, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[22] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. VIPS: A vision-based page segmentation algorithm. Technical Report MSR-TR-2003-79, Microsoft, 2003.

[23] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6), 1986.

[24] Jiuxin Cao, Bo Mao, and Junzhou Luo. A segmentation method for web page analysis using shrinking and dividing. *International Journal of Parallel, Emergent and Distributed Systems*, 25(2):93–104, 2010.

[25] Kenneth R. Castleman. *Digital Image Processing*, chapter 18. Prentice Hall, 1996.

[26] F. Cesarini, M. Gori, S. Marinai, and G. Soda. Structured document segmentation and representation by the modified x-y tree. In *Document Analysis and Recognition, 1999. ICDAR '99. Proceedings of the Fifth International Conference on*, pages 563–566, Sep 1999.

[27] Tony F. Chan and Luminita A. Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 10(2), 2001.

[28] Jinlin Chen, Ping Zhong, and T. Cook. Detecting web content function using generalized hidden markov model. In *5th Int. Conf. on Machine Learning and Applications*, pages 279–284, Dec 2006.

[29] Y. Chen, W. Ma, and H. Zhang. Detecting web page structure for adaptive viewing on small form factor devices. In *Proceedings of the Twelfth International World Wide Web Conference*, 2003.

[30] Yu Chen, Xing Xie, Wei-Ying Ma, and Hong-Jiang Zhang. Adapting web pages for small-screen devices. *IEEE Internet Computing*, 9(1):50–56, January 2005.

[31] Michael Cormier. Computer vision-based analysis of web page structure for assistive interfaces. In *Proceedings of the 13th Web for All Conference*, pages 1–2, 2016.

[32] Michael Cormier, Robin Cohen, Richard Mann, Kamal Rahim, and Donglin Wang. A robust vision-based framework for screen readers. In *2014 Workshop on Assistive Computer Vision and Robotics*, 2014.

[33] Michael Cormier, Richard Mann, Robin Cohen, and Karyn Moffatt. Classification via hidden markov trees for a vision-based approach to conveying webpages to users with assistive needs. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence*, 2016.

[34] Michael Cormier, Richard Mann, Karyn Moffatt, and Robin Cohen. Towards an improved vision-based web page segmentation algorithm. In *Proceedings of the 2017 Conference on Computer and Robot Vision (CRV)*, 2017.

[35] Michael Cormier, Karyn Moffatt, Robin Cohen, and Richard Mann. Purely vision-based segmentation of web pages for assistive technology. *CVIU special issue on Assistive Computer Vision and Robotics*, 2016.

[36] Matthew S Crouse, Robert D Nowak, and Richard G Baraniuk. Wavelet-based statistical signal processing using hidden markov models. *IEEE Trans. on Signal Processing*, 46(4):886–902, 1998.

[37] Will Crowther and Don Woods. Adventure. (Software), 1975–1976.

[38] Peter De Souza. Edge detection using sliding statistical tests. *Computer vision, graphics, and image processing*, 23(1):1–14, 1983.

[39] Ciro D'Elia, Giovanni Poggi, and Giuseppe Scarpa. A tree-structured markov random field model for bayesian image segmentation. *IEEE Transactions on Image Processing*, 12(10):1259–1273, 2003.

[40] Anna Dickinson, Roos Eisma, and Peter Gregor. The barriers that older novices encounter to computer use. *Universal Access in the Information Society*, 10(3):261–266, 2011.

[41] Morgan Dixon and James Fogarty. Prefab: Implementing advanced behaviors using pixel-based reverse engineering of interface structure. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1525–1534, 2010.

[42] Piotr Dollár and C Lawrence Zitnick. Structured forests for fast edge detection. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1841–1848. IEEE, 2013.

[43] James H Elder and Steven W Zucker. Local scale control for edge detection and blur estimation. *IEEE Transactions on pattern analysis and machine intelligence*, 20(7):699–716, 1998.

[44] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2009 (VOC2009) Results. http://www.pascal-network.org/challenges/VOC/voc2009/workshop/index.html.

[45] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results. http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html.

[46] Peter Fairweather and Shari Trewin. Cognitive impairments and web 2.0. *Universal Access in the Information Society*, 9:137–146, 2010.

[47] Peter G. Fairweather. How older and younger adults differ in their approach to problem solving on a complex website. In *Proceedings of the 10th International ACM SIGACCESS Conference on Computers and Accessibility*, Assets '08, pages 67–72, New York, NY, USA, 2008. ACM.

[48] Robert Douglas Ferguson, Michael Massimi, Emily Anne Crist, and Karyn Anne Moffatt. Craving, creating, and constructing comfort: insights and opportunities for technology in hospice. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*, pages 1479–1490. ACM, 2014.

[49] Clément Fredembach, Michael Schröder, and Sabine Süsstrunk. Eigenregions for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(12), 2004.

[50] G.W. Furnas. Generalized fisheye views. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'86)*, pages 16–23, 1986.

[51] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, and Jose Garcia-Rodriguez. A review on deep learning techniques applied to semantic segmentation. *arXiv preprint arXiv:1704.06857*, 2017.

[52] Carole Goble, Simon Harper, and Robert Stevens. The travails of visually impaired web travellers. In *Proceedings of the eleventh ACM on Hypertext and hypermedia*, pages 1–10. ACM, 2000.

[53] Google. ChromeVox. http://www.chromevox.com/. Accessed 2014-03-16.

[54] U. Grenander and A. Srivastava. Probability models for clutter in natural images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(4):424–429, Apr 2001.

[55] J. Hammersly and P. Clifford. Markov fields on finite graphs and lattices. Unpublished manuscript, 1971.

[56] Peter JB Hancock, Roland J Baddeley, and Leslie S Smith. The principal components of natural images. *Network: computation in neural systems*, 3(1):61–70, 1992.

[57] Vicki L. Hanson. Cognition, age, and web browsing. In Constantine Stephanidis, editor, *Universal Access in Human-Computer Interaction. Addressing Diversity*, volume 5614 of *Lecture Notes in Computer Science*, pages 245–250. Springer Berlin Heidelberg, 2009.

[58] V.L. Hanson, J.P. Brezin, S. Crayne, S. Keates, R. Kjeldsen, J.T. Richards, C. Swart, and S. Trewin. Improving web accessibility through an enhanced open-source browser. *IBM Systems Journal*, 44:573–588, 2005.

[59] S. G. Hart and L. E. Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In P. A. Hancock and N. Meshkati, editors, *Human Mental Workload*. North Holland Press, Amsterdam, 1988.

[60] T. A. Hart, B. S. Chaparro, and C. G. Halcomb. Evaluating websites for older adults: adherence to senior-friendly guidelines and end-user performance. *Behaviour & Information Technology*, 27(3):191–199, 2008.

[61] F. Hassan and A. H. Ahmed. Web document segmentation for better extraction of information: A review. *International Journal of Computer Applications*, 110(3), 2015.

[62] D Hawthorn. Possible implications of aging for interface designers. *Interacting with Computers*, 12(5):507, 2000.

[63] Xuming He, Richard S Zemel, and Miguel Á Carreira-Perpiñán. Multiscale conditional random fields for image labeling. In *Computer vision and pattern recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE computer society conference on*, volume 2. IEEE, 2004.

[64] Michael D Heath, Sudeep Sarkar, Thomas Sanocki, and Kevin W Bowyer. A robust visual method for assessing the relative performance of edge-detection algorithms.

IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(12):1338–1359, 1997.

[65] Thomas J Hebert and Denis Malagre. Optimized edge detection using a priori models. In *Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference*, volume 1, pages 303–307. IEEE, 1994.

[66] Ian Hickson, Tantek Çelik, Håkon Wium Lie, and Bert Bos. Cascading style sheets level 2 revision 1 (CSS 2.1) specification. W3C recommendation, W3C, June 2011. http://www.w3.org/TR/2011/REC-CSS2-20110607/.

[67] Andreas Holzinger. Interactive machine learning for health informatics: when do we need the human-in-the-loop? *Brain Informatics*, 3(2):119–131, 2016.

[68] Yili Hong. On computing the distribution function for the poisson binomial distribution. *Computational Statistics & Data Analysis*, 59:41 – 51, 2013.

[69] Kasper Hornbæk and Effie Lai-Chong Law. Meta-analysis of correlations among usability measures. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 617–626. ACM, 2007.

[70] Paul VC Hough. Method and means for recognizing complex patterns, December 18 1962. US Patent 3,069,654.

[71] Faustina Hwang, Nic Hollinworth, and Nitin Williams. Effects of target expansion on selection performance in older computer users. *ACM Trans. Access. Comput.*, 5(1):1:1–1:26, September 2013.

[72] A. Jepson and R. Mann. Qualitative probabilities for image interpretation. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1123–1130 vol.2, 1999.

[73] Eun Hwa Jung and S Shyam Sundar. Senior citizens on facebook: How do they interact and why? *Computers in Human Behavior*, 61:27–35, 2016.

[74] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331, 1988.

[75] Hanhwe Kim and Stephen C. Hirtle. Spatial metaphors and disorientation in hypertext browsing. *Behaviour & Information Technology*, 14(4):239–250, 1995.

[76] Koichi Kise. *Handbook of Document Image Processing and Recognition*, chapter 5: Page Segmentation Techniques in Document Analysis. Springer-Verlag, London, 2014.

[77] Alfred Kobsa and Wolfgang Wahlster, editors. *User Models in Dialog Systems*. Springer-Verlag, 1989.

[78] S. Konishi, A. L. Yuille, J. M. Coughlan, and Song Chun Zhu. Statistical edge detection: learning and evaluating edge cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(1):57–74, 2003.

[79] Milos Kovacevic, Michelangelo Diligenti, Marco Gori, and Veljko Milutinovic. Recognition of common areas in a web page using visual information: a possible application in a page classification. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 250–257. IEEE, 2002.

[80] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in neural information processing systems*, pages 109–117, 2011.

[81] Bernhard Krüpl-Sypien, Ruslan R. Fayzrakhmanov, Wolfgang Holzinger, Mathias Panzenböck, and Robert Baumgartner. A versatile model for web page representation, information extraction and content re-packaging. In *Proceedings of the 11th ACM Symposium on Document Engineering*, DocEng '11, pages 129–138, New York, NY, USA, 2011. ACM.

[82] Malay Kumar Kundu and Sankar K Pal. Thresholding for edge detection using human psychovisual phenomena. *Pattern Recognition Letters*, 4(6):433–441, 1986.

[83] Sri Hastuti Kurniawan, Alasdair King, David Gareth Evans, and PL Blenkhorn. Personalising web page presentation for older people. *Interacting with computers*, 18(3):457–477, 2006.

[84] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning 2001 (ICML 2001)*, pages 282–289, 2001.

[85] Alison Lee. Scaffolding visually cluttered web pages to facilitate accessibility. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '04, pages 90–93, New York, NY, USA, 2004. ACM.

[86] Ke Li, Bharath Hariharan, and Jitendra Malik. Iterative instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3659–3667, 2016.

[87] Hubert W Lilliefors. On the kolmogorov-smirnov test for normality with mean and variance unknown. *Journal of the American statistical Association*, 62(318):399–402, 1967.

[88] David G Lowe. Perceptual organization and visual recognition. Technical report, DTIC Document, 1984.

[89] J.U. Mahmud, Y. Borodin, and I.V. Ramakrishnan. Csurf: a context-driven non-visual webbrowser. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007*, pages 31–40, New York, 2007. ACM.

[90] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[91] Jiri Matas, Charles Galambos, Josef Kittler, et al. Progressive probabilistic hough transform. 1998.

[92] Karyn Moffatt. Older-adult HCI: Why should we care? *Interactions*, 20(4):72–75, July 2013.

[93] Karyn Moffatt, Jessica David, and Ronald M. Baecker. *Connecting Grandparents and Grandchildren*, pages 173–193. Springer London, London, 2013.

[94] David Mumford and Agnès Desolneux. *Pattern Theory*. A K Peters, Ltd., Natick, MA, 2010.

[95] David Mumford and Jayant Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on pure and applied mathematics*, 42(5):577–685, 1989.

[96] G. Nagy and S. Seth. Hierarchical representation of optically scanned documents. In *Proc. of ICPR*, pages 347–349, 1984.

[97] Nvda 2015.4 user guide. `www.nvaccess.org/files/nvda/documentation/userGuide.html` (accessed Feb. 15, 2016).

[98] Bruno A Olshausen and David J Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607, 1996.

[99] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulation. *J. Comput. Phys.*, 79:12–49, 1988.

[100] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[101] Ofir Pele and Michael Werman. A linear time histogram metric for improved sift matching. In *ECCV*, 2008.

[102] Ofir Pele and Michael Werman. Fast and robust earth mover's distances. In *ICCV*, 2009.

[103] Georgios Petasis, Pavlina Fragkou, Aris Theodorakos, Vangelis Karkaletsis, and Constantine D Spyropoulos. Segmenting HTML pages using visual and semantic information. In *Workshop Programme*, page 18, 2008.

[104] Taciana Pontual Falcão, Xiaofeng Yong, Elisabeth Sulmont, Robert Douglas Ferguson, and Karyn Moffatt. A digital pen and paper email system for older adults. In *Adjunct Publication of the 30th Annual ACM Symposium on User Interface Software and Technology*, pages 189–191. ACM, 2017.

[105] MB Priestley and MT Chao. Non-parametric function fitting. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 385–392, 1972.

[106] R. R. Rakesh, P. Chaudhuri, and C. A. Murthy. Thresholding in edge detection: a statistical approach. *IEEE Transactions on Image Processing*, 13(7):927–936, 2004.

[107] John T. Richards and Vicki L. Hanson. Web accessibility: A broader view. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 72–79, New York, NY, USA, 2004. ACM.

[108] Jonathan Robie, Lauren Wood, Steven B Byrne, Philippe Le Hégaret, Arnaud Le Hors, Mike Champion, and Gavin Nicol. Document object model (DOM) level 2 core specification. W3C recommendation, W3C, November 2000. http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113/.

[109] Romisa Rohani Ghahari, Mexhid Ferati, Tao Yang, and Davide Bolchini. Back navigation shortcuts for screen reader users. In *Proc. 14th Int. ACM SIGACCESS conf. on Computers and Accessibility*, pages 1–8. ACM, 2012.

[110] Justin Romberg, Hyeokho Choi, Richard Baraniuk, and N Kingbury. Multiscale classification using complex wavelets and hidden markov tree models. In *Image Processing, 2000. Proceedings. 2000 International Conference on*, volume 2, pages 371–374. IEEE, 2000.

[111] Justin K Romberg, Hyeokho Choi, and Richard G Baraniuk. Bayesian tree-structured image modeling using wavelet-domain hidden markov models. *IEEE Transactions on image processing*, 10(7):1056–1068, 2001.

[112] Amir Rosenfeld, Markus D. Solbach, and John K. Tsotsos. Totally looks like - how humans compare, compared to machines. *CoRR*, abs/1803.01485, 2018.

[113] Ruth Rosenholtz, Yuanzhen Li, Jonathan Mansfield, and Zhenlan Jin. Feature congestion: A measure of display clutter. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, pages 761–770, New York, NY, USA, 2005. ACM.

[114] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000.

[115] Eric Saund and Thomas P Moran. A perceptually-supported sketch editor. In *Proceedings of the 7th annual ACM symposium on User interface software and technology*, pages 175–184. ACM, 1994.

[116] Anrés Serna and Beatriz Marcotegui. Detection, segmentation and classification of 3d urban objects using mathematical morphology and supervised learning. *ISPRS Journal of Photogrammetry and Remote Sensing*, 93:243–255, 2014.

[117] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[118] F.Y. Shih, S.-S. Chen, D.C.D. Hung, and P.A. Ng. A document segmentation, classification and recognition system. In *Proceedings of the Second International Conference on Systems Integration, 1992 (ICSI '92)*, pages 258–267, Jun 1992.

[119] Kristen Shinohara and Jacob O. Wobbrock. In the shadow of misperception: assistive technology use and social interactions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*, pages 705–714, New York, NY, USA, 2011. ACM.

[120] Ruihua Song, Haifeng Liu, Ji-Rong Wen, and Wei-Ying Ma. Learning important models for web page blocks based on layout and content analysis. *SIGKDD Explor. Newsl.*, 6(2):14–23, December 2004.

[121] H. Takagi, C. Asakawa, K. Fukuda, and J. Maeda. Site-wide annotation: Reconstructing existing pages to be accessible. In *ASSETS 2002*, pages 81–88. ACM Press, 2002.

[122] Shari Trewin, John T. Richards, Vicki L. Hanson, David Sloan, Bonnie E. John, Cal Swart, and John C. Thomas. Understanding the role of age and fluid intelligence in information search. In *Proceedings of the 14th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '12, pages 119–126, New York, NY, USA, 2012. ACM.

[123] Thomas S. Tullis. Older adults and the web: lessons learned from eye-tracking. In Constantine Stephanidis, editor, *Proceedings of the 4th international conference on Universal access in human computer interaction: coping with diversity (UAHCI'07)*, Berlin, Heidelberg, 2007. Springer-Verlag.

[124] Matthew A Turk and Alex P Pentland. Face recognition using eigenfaces. In *Proc. Computer Vision and Pattern Recognition*, pages 586–591. IEEE, 1991.

[125] TUWIEN Database and Artificial Intelligence Group. TUWIEN Project ABBA: Web Accessibility. http://www.dbai.tuwien.ac.at/proj/ABBA/. Accessed June 4, 2014.

[126] Olga Veksler. Star shape prior for graph-cut image segmentation. In *European Conference on Computer Vision*, pages 454–467. Springer, 2008.

[127] Rüdiger von der Heydt, Esther Peterhans, and Gunter Baumgartner. Illusory contours and cortical neuron responses. *Science*, 224(4654):1260–1262, 1984.

[128] W3C. WAI-ARIA 1.0 primer: An introduction to rich internet application accessibility challenges and solutions. https://www.w3.org/TR/wai-aria-primer/ (accessed Feb 16, 2016), 2010.

[129] W3C. Accessible rich internet applications (WAI-ARIA) 1.0. w3c recommendation. http://www.w3.org/TR/wai-aria (accessed Feb 16, 2016), 2014.

[130] Martin J Wainwright, Tommi S Jaakkola, and Alan S Willsky. A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory*, 51(7):2313–2335, 2005.

[131] Masahiro Watanabe, Daisuke Asai, and Yoko Asano. Improving accessibility through the visual structure of web contents. In *Proceedings of the 4th International Conference on Universal Access in Human-Computer Interaction (UAHCI '07)*, pages 185–192, 2007.

[132] Tingting Wei, Yonghe Lu, Xuanjie Li, and Jinglun Liu. Web page segmentation based on the hough transform and vision cues. In *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2015 Asia-Pacific*, pages 865–872. IEEE, 2015.

[133] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(11):1101–1113, 1993.

[134] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.

[135] Yi Yang, Sam Hallman, Deva Ramanan, and Charless C. Fowlkes. Layered object models for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1731–1743, 2012.

[136] Y. Yesilada, S. Harper, C.A. Goble, and R. Stevens. Screen readers cannot see (ontology based semantic annotation for visually impaired web travellers). In N. Koch, P. Fraternali, and M. Wirsing, editors, *ICWE 2004*, volume 3140 of *LNCS*, Heidelberg, 2004. Springer.

[137] Y. Yesilada, R. Stevens, S. Harper, and C. Goble. Evaluating dante: Semantic transcoding for visually disabled users. *ACM Trans. Hum. Comput. Interact.*, 14:66–96, 2007.

[138] Xinyi Yin and Wee Sun Lee. 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 '05, pages 1150–1151, New York, NY, USA, 2005. ACM.

[139] Jan Zeleny, Radek Burget, and Jaroslav Zendulka. Box clustering segmentation: A new method for vision-based web page preprocessing. *Information Processing & Management*, 53(3):735–750, 2017.

[140] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1529–1537, 2015.

[141] Xiaolong Zhu, Huijing Zhao, Yiming Liu, Yipu Zhao, and Hongbin Zha. Segmentation and classification of range image from an intelligent vehicle in urban environment. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1457–1462. IEEE, 2010.

[142] AA Zlatopolsky. Automated document segmentation. *Pattern Recognition Letters*, 15(7):699–704, 1994.

Appendix A

Segmentation Algorithm

Algorithm SegmentRegion shows our original segmentation algorithm, including the optimization process, as described in Section 3.2. It produces a tree, with the key of each node providing the necessary information about the division into child regions; this is represented by a tuple of the form (K, C), where K is the key and C is the set of children. Parameters of the segmentation algorithm which are not generally altered are the maximum segmentation depth d_{max}, the maximum number of candidate boundaries (either horizontal or vertical) n_{max}, the threshold of significance update increment δ_t, and the initial threshold of significance t_{init}. These parameters are set based on performance considerations [1]. The "current depth" parameter d represents the depth of the tree if generation stops at the current iteration; in calling the function, it is set to 1, and it is updated automatically for recursive calls.

[1] In our implementation, $d_{max} = 6$, $\delta_t = 0.5$, and $t_{init} = 3$. We pre-computed sets of tilings for candidate boundary grids of different sizes for efficiency reasons; rather than a fixed n_{max}, the maximum number of grid cells formed by the candidate boundaries was 35, except for single columns or rows, where it was somewhat lower at 23. The algorithm shown is slightly idealized, using the simpler (but possibly slower) fixed maximum.

Input: Region image I, current depth d
Output: Segmentation tree or subtree
Calculate pixel-wise horizontal and vertical edge probabilities E_h, E_v (Equation 3.7);
Let $t \leftarrow t_{init}$ be the significance threshold for peak detection;
foreach *row* i **do** $H_i \leftarrow \prod_j E_h(i,j)$;
foreach *column* j **do** $V_j \leftarrow \prod_i E_v(i,j)$;
repeat
  $HPeaks \leftarrow \{r : E_h(r) > \text{mean}(E_h) + t\text{stddev}(E_h)\}$;
  $VPeaks \leftarrow \{c : E_v(c) > \text{mean}(E_v) + t\text{stddev}(E_v)\}$;
  $t \leftarrow t + \delta_t$;
until $|VPeaks| < n_{max} \wedge |HPeaks| < n_{max}$;
if $(HPeaks == \emptyset \wedge VPeaks = \emptyset) \vee d > d_{max}$ **then return** $(\emptyset, \emptyset)$;
$q^* \leftarrow 0, S^* \leftarrow \emptyset$;
Let T represent the search space of tilings with divisions in $HPeaks$ and $VPeaks$;
foreach *tiling* $S \in T$ **do**
  $q \leftarrow \text{SegmentationQuality}(S, E_h, E_v)$;
  if $q > q^*$ **then** $q^* \leftarrow q, S^* \leftarrow S$;
end
foreach *region* $R \in S^*$ **do** $C_R = \text{SegmentRegion}(\text{Image}(R), d+1)$;
return $(S^*, \{C_R : R \in S^*\})$

Function SegmentRegion(I, d): Recursive segmentation algorithm, including optimizing over possible segmentations of each region.

Appendix B

User Study Materials

In this appendix we show, for reference, the materials shown to participants in the user study in Chapter 5. In Section B.1, we show an example script for explaining the procedures of the experiment to a participant. In Section B.2, we show the surveys presented to users to elicit their subjective impressions of the assistive and "vanilla" interfaces.

In this study, the participants viewed these interfaces on a 22-inch monitor with a resolution of 1680×1050. The viewable area of the page was 1024×768 pixels in size. The experiment took place in a well-lit room. The participants sat in an unmodified office chair at an unmodified desk, and were free to sit in a comfortable position.

B.1 Introduction Script

The following script was the basis of the introduction given to participants to the procedures of the experiment. In many cases the script was modified to respond to participants' questions.

> Good morning/afternoon, [participant]. In this experiment, assuming you choose to participate, you will be browsing through a web page to find a specific article. The web pages will be presented in two different ways: one "plain", as it would usually be presented, and another with features to highlight and magnify selected regions of the page.
>
> The first step in the experiment is to fill in a very brief background survey about your web experience and preferences. This is followed by an opportunity to practice with and configure the interface that highlights and magnifies

regions. Clicking on a part of the web page moves the highlighted region to the place you clicked. The size of the region can be controlled by the slider on the right. At this point you will be able to set the degree of magnification and highlighting that you will use in the remainder of the experiment, also using the sliders on the right.

After these preliminary steps, you will begin the main part of the experiment. This is divided into two stages, one for each type of interface. In each of these stages, you will perform a series of tasks in which the objective is to find an article (which will be described on the screen) in an example of a news web page as quickly as possible. Once you've found the target article, click somewhere on the article or description, then click th Done button on the right. After completing each stage, you will be presented with a brief questionaire about your experience using the interface.

Do you have any questions?

Please review these two documents and, if you decide that you would like to participate, sign the consent form.

B.2 Surveys

In this section we show the surveys administered to participants in the course of the experiment. We use screenshots to depict them as the participants saw them, including all options.

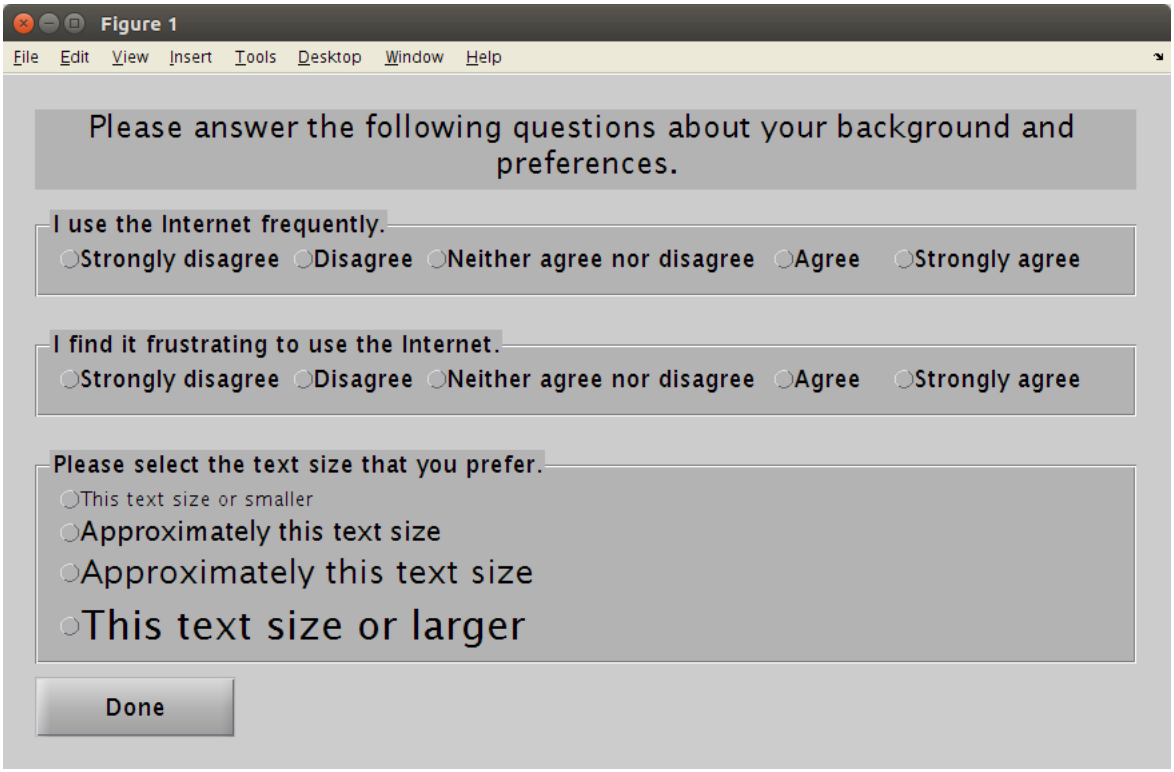


Figure B.1: Background questionnaire administered to participants prior to beginning the main tasks of the experiment. This survey is used to establish participants' computer experience and preferences for text size, which may influence the desired level of magnification.

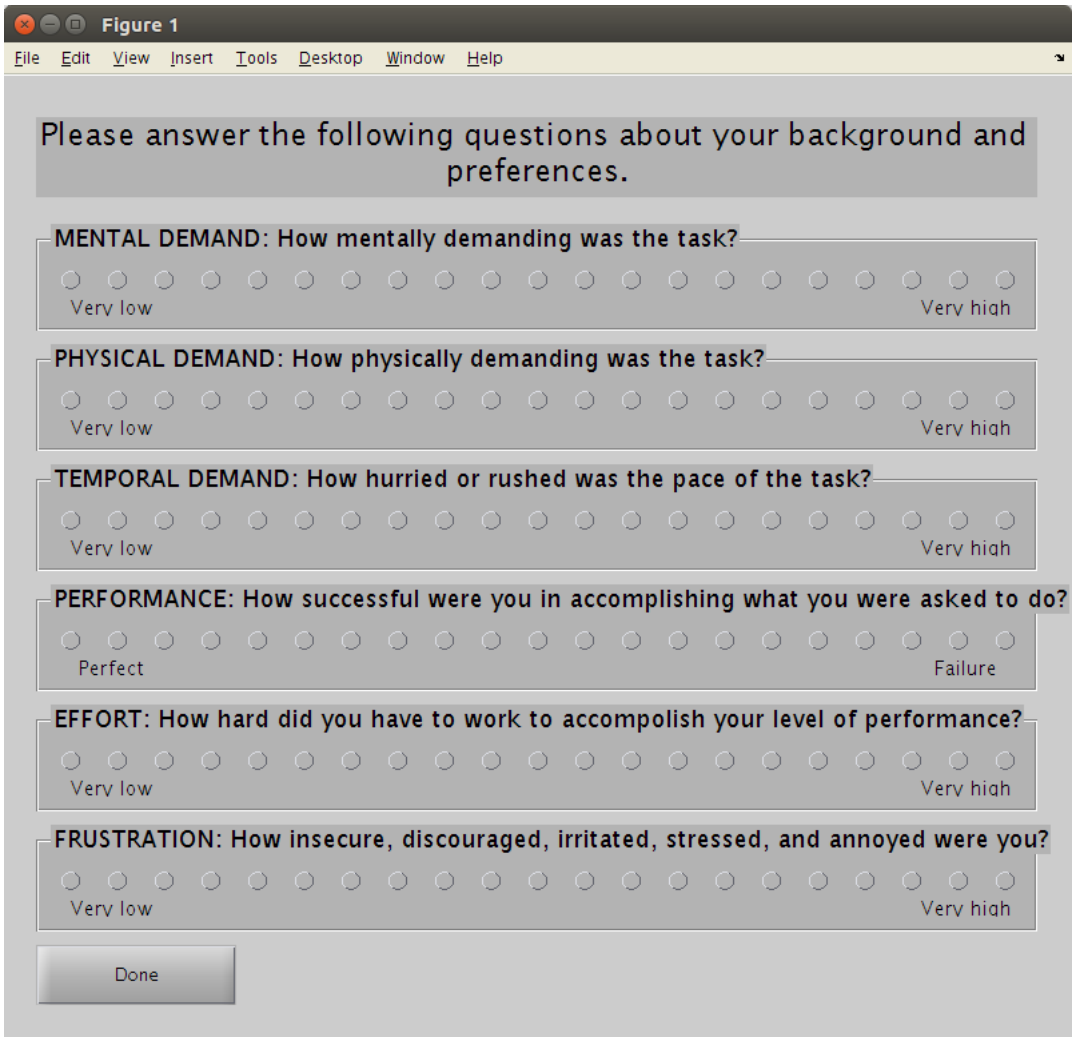


Figure B.2: NASA TLX survey administered to participants after completing each series of tasks. One series of tasks is performed using the assistive interface, and one is performed using the "vanilla" interface without assistive features. The results of this survey are used to determine which interface is preferable on each measure.

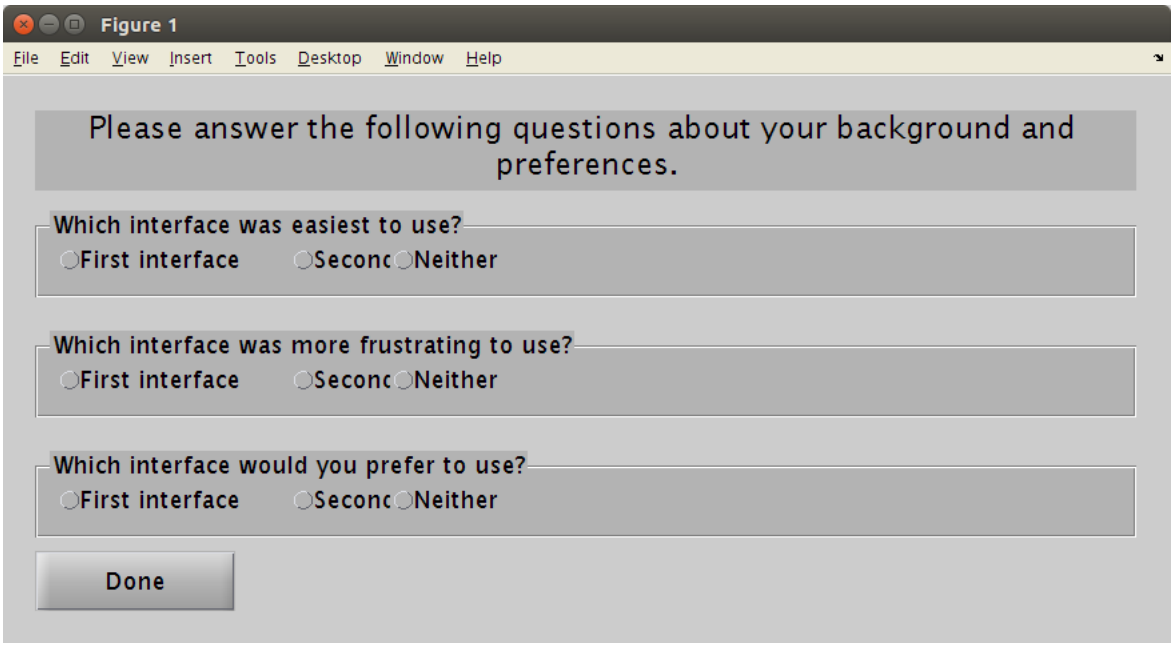


Figure B.3: Preference questionnaire administered to participants after completing all tasks with both the assistive and vanilla interfaces to determine which interface was subjectively preferable.

Appendix C

Extended Computer Vision Discussion

This appendix returns to the related work on computer vision segmentation and classification covered in Section 2.5, a section which also included important contrast with our own approach. We now present that material in greater detail and introduce additional references. This provides a deeper discussion of the literature, leading to greater insights into how we situate with respect to the field.

C.1 Edge Detection

Edge detection is one of the key low-level problems in computer vision. Many higher-level algorithms assume the existence of some form of edge detection or include an edge detection method suitable to their needs in the algorithm. Snakes, active contours, and level set methods are a good example of such algorithms, which perform higher-level detection of semantic contours and segmentation based on low-level edge detection (see Section C.2.1).

Perhaps the most famous paper in edge detection is Canny's 1986 "A Computational Approach to Edge Detection" [23], which systematically defined criteria for optimal filters for detection of arbitrary edge types, focusing on the development of a step-edge detector. Canny defined three principal criteria for a good edge detection algorithm:

- **Good detection** requires that the edge detection method have a low probability of false negatives and a low probability of false positives; a perfect edge detector would report an edge if and only if an edge exists.

- **Good localization** requires that the detected edges correspond well in location to the true location of the edge; a perfect edge detector would place each detection exactly on the center of the edge in the image.
- **Single detection** requires that edges in the image do not produce multiple edge detector responses; a perfect edge detector would detect each edge in the image exactly once.

These qualitative criteria were then formulated mathematically for one-dimensional signals. Canny used these definitions to implicitly define an optimal filter for step edges. Although analytical solution was not feasible, numerical solutions were found. The first derivative of a Gaussian, which is much more tractable in practice that the optimal numerical solution, was found to be nearly optimal.

In order to produce hard edge detections from the continuous filter response, thresholding is necessary. Using a single global threshold tends to produce broken edges due to noise intermittently bringing the response below the threshold along the length of an edge in the image. Instead, Canny recommended using hysteresis: points where the detector response is above a high threshold are accepted immediately, as are connected points above a low threshold. Filter width was also considered, and found to involve a tradeoff between reliability of detection and accuracy of localization. In addition to the development of optimal one-dimensional filters, Canny discussed directional filters for two-dimensional images and showed there usefulness in improving performance [23]. The Canny edge detector has remained a mainstay of edge detection to this day.

Adaptive thresholds for edge detection are critical to the performance of many algorithms. Heath *et al.* [64], for example, demonstrated that selecting parameters for edge detection algorithms on a per-image basis provides a significant improvement in performance, as rated by human judges, for several common edge detectors. Even this, however, is not sufficient; as pointed out by Kundu and Pal [82], from the perspective of human psychophysical contraints, and by Rakesh *et al.* [106], from the perspective of statistical considerations (among others), local adaptation of thresholds is required. In our own work, we use the empirical distributions of edge strength on either side of a proposed line to adapt to local image properties (see Sections 3.2.2 and 3.3.1).

The locally adaptive thresholding technique described by Rakesh *et al.* [106] also uses statistical methods. More specifically, this method calculates a test statistic $S(x, y)$ for each pixel (x, y) and thresholds the test statistic using hysteresis with upper and lower thresholds. The test statistic is based upon the partial derivatives of a version of the image smoothed by a Priestley-Chao kernel smoother [105]. For an $m \times n$-pixel image Z, the

smoothed image f is defined by

$$f(x,y) = \frac{1}{2\pi mnh^2} \sum_{i=1}^{m} \sum_{j=1}^{n} K\left(\frac{x-i}{h}\right) K\left(\frac{y-j}{h}\right) Z(i,j) \tag{C.1}$$

where h is a smoothing parameter and K is a kernel function. The partial derivative of f with respect to x is

$$f_x(x,y) = \frac{1}{2\pi mnh^2} \sum_{j=1}^{n} a_{j,h}(y) \left(\sum_{i=1}^{m} a'_{i,h}(x) Z(i,j) \right) \tag{C.2}$$

where $a_{i,h} = K(\frac{x-i}{h})$. The definition of $f_y(x,y)$ is analogous. For each pixel (x,y), there is a covariance matrix for the two components of the gradient, defined to be

$$\Sigma(x,y) = \begin{bmatrix} \sigma_{11}(x,y) & \sigma_{12}(x,y) \\ \sigma_{12}(x,y) & \sigma_{22}(x,y) \end{bmatrix} \tag{C.3}$$

The test statistic $S(x,y)$ is defined in terms of this covariance matrix:

$$S(x,y) = [f_x(x,y), f_y(x,y)]\Sigma^{-1}(x,y)[f_x(x,y), f_y(x,y)]^T \tag{C.4}$$

$$= \frac{f_x^2(x,y)\sigma_{22}(x,y) + f_y^2(x,y)\sigma_{11}(x,y) - 2f_x(x,y)f_y(x,y)\sigma_{12}(x,y)}{\sigma_{11}(x,y)\sigma_{22}(x,y) - \sigma_{12}^2(x,y)} \tag{C.5}$$

The covariance-based approach, like our method for estimating the probability that an edge is locally significant, uses the local distribution of estimated gradients to determine whether an edge is sufficiently strong relative to its neighbourhood to be detected as an edge. There are a number of differences in details, such as the use of a soft kernel-based neighbourhood by Rakesh *et al.*, but the principal differences are that our method uses Bayesian statistics to produce an estimate of the *probability* that an edge is locally significant, rather than producing a score which is then thresholded, and that our method uses a nonparametric estimate of the complete distribution of edge strengths rather than using summary statistics such as the covariance matrix. We believe that the use of a nonparametric representation of the distribution of edge strengths is advantageous, especially in our domain where the distribution is very far from a Gaussian.

Statistical models of image properties can also be used in edge detection. Elder and Zucker proposed [43] a model for determining the minimum local scale at which a gradient detection can be assumed (with a specified level of confidence) to result from an edge in the image, rather than from sensor noise, based on a parametric model of the noise of a

given camera. An early example of statistical modelling of regions as a strategy for edge detection appears in a paper [38] by deSouza. This paper considers a one-dimensional profile, taken as a slice from an image, and estimates the edge strength by applying a statistical test to determine if there is a significant difference between two halves of a sliding window. If the two halves are statistically different, this indicates the presence of an edge at the center of the window. To account for the possibility that the two regions are nonuniform, deSouza's algorithm parametrically models the two halves of the window using second-order polynomial regression and performs the test based upon the constant terms. For textured regions, deSouza proposed using the ratio of the variances of the two halves of the window, or using a χ^2 test on autoregressive models of the two halves of the window. DeSouza warns, however, that while the results of the statistical test can be used to indicate edge strength, it cannot necessarily be taken as a valid hypothesis test, since the assumptions made by the procedure are not always satisfied in this domain.

Our system uses a different approach to statistical edge detection. Rather than comparing a pair of windows in the image, we compare a single measurement of preliminary edge strength to a pair of adjacent neighbourhoods, using a nonparametric representation of the distributions of preliminary edge strengths in these regions. Our approach is intended specifically to find the edges of the texture elements (such as characters in a paragraph of text) at the edges of texture regions; in the domain of rendered web pages, it is reasonable to assume (due to the rendering process) that these edge segments will be very precisely aligned.

A Bayesian approach to edge detection was described by Hebert and Malagre in [65], using a Gibbs distribution (see Section C.2.3) to represent *a priori* intuitions about the plausible structure of an edge image. A plausible line image is assumed to consist of continuous lines one pixel in width, with no isolated edge pixels and a minimum separation of two pixels between parallel edges. The prior probability distribution over small edge image patches is uniformly zero for those which do not satisfy these constraints, and uniformly positive for those which do. Incorporating evidence for a simple edge detector (such as a gradient filter) allows the calculation of a posterior probability for each patch; optimizing over these allows the calculation of a maximum *a posteriori* estimate of the true edge image.

One example of edge detection by learning the appearance of edges is the system proposed by Konishi *et al.* [78]. In this approach, images were processed with filter banks to produce vectors of filter responses for each pixel. Using ground truth labels, the joint distributions of filter responses for edge and non-edge pixels were learned. The filter banks included conventional filters such as the Laplacian operator, and filters such as the Gabor filter which are inspired by the human visual system. The joint distributions were repre-

sented nonparametrically, using histograms. To reduce the sizes of these high-dimensional histograms, the bins were allowed to adapt to the data. Using these joint distributions, it was possible to predict the log-likelihood ratio

$$\log \frac{\Pr(\phi(I(x))|x \text{ is on an edge})}{\Pr(\phi(I(x))|x \text{ is not on an edge})} \tag{C.6}$$

where x is a pixel in the image I and ϕ represents the filter bank. The log-likelihood ratio can simply be thresholded to produce an edge map; alternatively, better performance can be obtained by using learned spatial cues. These cues were determined by applying a filter bank to a map of the posterior probability of the presence of an edge. Using this procedure, properties such as non-maximum suppression and hysteresis emerged from the data. Interestingly, using spatial cues produced a small quantitative improvement n performance, but a large qualitative improvement.

Like our approach to edge detection, Konishi *et al.* used a nonparametric model to estimate edge probabilities. In the case of our algorithm, however, this distribution was derived solely from the neighbourhood of the pixel in question, rather than being learned from ground truth data as in [78]. This is an important difference; our algorithm is based on *a priori* assumptions about the properties of the domain, while the algorithm described by Konishi *et al.* focuses entirely on learning from the data. Our assumptions proved effective in our domain, although it may be interesting to test a learning-based approach similar to those in [42] or [78] in the domain of web pages (see Section 8.3.2).

As shown by these examples from the literature, our approach to statistical edge detection is part of a group of related statistical approaches to the problem of edge detection. Our algorithm is specifically designed to address the expected characteristics of semantically significant edges in the domain of rendered web pages. As such it differs in details from the related statistical approaches, but shares a common emphasis of using local image statistics to adapt to spatially varying characteristics of the image in order to produce reliable edge information even when the image is cluttered or otherwise challenging.

C.2 Image Segmentation

The problem of image segmentation is, put simply, the problem of partitioning an image into semantically meaningful regions. It is a complex and ill-posed problem. In many cases, multiple different segmentations can be considered "correct". A remarkable range of approaches have been taken to image segmentation; a complete survey of the development

of the field would require an entire book. Here we present a discussion of selected work in this area, including papers of particular relevance to the developments presented in this thesis.

C.2.1 Edges and Active Contours

Our page segmentation method uses edges as the primary evidence for segmenting a page. This approach to image segmentation has a long history in computer vision. It is an intuitively appealing approach; it is reasonable to expect that semantically significant regions are likely to have a different appearance from their surroundings, and this will often create strong edges along the border between regions. The borders of regions are one way of defining a segmentation of the image, so edges are an appealing form of evidence to examine.

A classic example of edge-based segmentation is the snake model proposed by Kass *et al.* in 1988 [74]. Snakes are dynamic splines, moving under physically-inspired forces to nearby semantically significant contours. They are not fully autonomous, but require initialization, either manually or by some other algorithm. There are three causes of these forces: first, internal forces representing the rigidity and resistance to bending of the spline; second, forces upon the spline imposed by the image; and third, forces imposed by external constraints provided by a user or by another algorithm. Formally, the energy of a snake is defined by the equation

$$E_s = \int_0^1 \left(E_v(v(s)) + E_I(v(s)) + E_C(v(s))\right) ds \tag{C.7}$$

where $v(s) = (x(s), y(s))$ is a parametric representation of the spline, E_v represents the internal energy of v, E_I represents the energy due to features in the image I, and E_C represents the energy due to external constraints. The internal energy, as defined by Kass *et al.*, is

$$E_v = \frac{\alpha(s)}{2}\left(|\frac{dv}{ds}|^2\right) + \frac{\beta(s)}{2}\left(|\frac{d^2v}{ds^2}|^2\right) \tag{C.8}$$

where α and β control the local importance of the first- and second-order terms, respectively. In a physical sense, the first-order term produces membrane-like behaviour, and the second-order term produces thin plate-like behaviour. In both cases, the rigidity of the spline has a regularizing effect. The external constraints, represented by E_C, take the form of spring-like attraction between a point on the spline and an arbitrary point, and $\frac{1}{r^2}$ repulsion from a point. The key feature of snakes that makes them a computer vision tool,

rather than a (probably rather amusing) physics toy, is the image term, E_I. In [74], the image term is the sum of three components, representing attraction to light or dark lines, attraction to edges, and attraction to line terminations and corners. The first of these components uses the value of the image at a position, the second uses the gradient of the image (or some other edge detector), and the third uses the curvature of level set lines. Snakes can be used to find open contours, acting as an edge detector, or closed contours, acting as a segmentation method.

Although the physical interpretation of snakes is a valuable property, the parametric nature of the spline also leads to some important limitations. One key limitation is that, barring a separate reparameterization step, the topology of the curve cannot change. This is very important in images where there are multiple foreground objects; the topology of an initial single contour must change to produce multiple separate regions. Similarly, regions with holes require topological changes if the initial contour does not have holes. To address this, some researchers turned to a technique proposed by Osher and Sethian [99] in the context of physical simulations. This technique, called the level set method, represents the region boundary implicitly, as the level set where a Lipschitz function on the image domain is equal to zero [99]. An image segmentation algorithm using the level set method simulates the evolution of this function rather than the evolution of the contour itself; topological changes occur naturally as the function drops below or rises above zero. This approach is naturally suited to active contour models, since it was originally designed to simulate boundaries whose local rate of propagation depends upon their curvature [99]. In image segmentation, the evolution of the function is designed to converge to the borders of objects. Broadly speaking, level set methods can be divided into those which use edges to define the borders of objects, and those which use the properties of areas inside and outside the boundary.

A level set approach to image segmentation, based upon region properties rather than edges, was proposed by Chan and Vese [27]. Although not an edge-based method (and therefore outside of the principal focus of this section), it is worth briefly discussing it as an example of area-based level set segmentation. Chan and Vese's model is based on a special case of the Mumford-Shah functional [95], in which the image is modelled as a piecewise-constant function, with one constant inside the boundary and another outside it. In the more general case of the Mumford-Shah functional, the image is modelled by piecewise smooth functions. Chan and Vese describe a method for finding a boundary (and corresponding constant values) that optimizes a piecewise constant fit to the image. The resulting energy function is nonconvex, but the authors describe techniques for finding a globally optimal solution regardless of the initial contour. One key advantage of this approach over those that use the image gradient to stop the evolution of the curve is that

it is not vulnerable to the contour moving through gaps in the edges around an object [27].

The active contour methods described here are related to our approach in that they produce segmentations supported by edges while imposing structural constraints on region shapes. The structural assumptions made by our method are, however, very different from those made by active contours. Because of the web page rendering process, we assume that the relevant edges are straight and axis-parallel, while most active contour methods penalize but allow strong curvature if it is supported by evidence in the image. Our segmentation system is also inherently hierarchical, and incorporates a prior probability distribution over segmentation trees.

C.2.2 Graph Cut Clustering

Graph cut clustering has historically been a prominent technique in image segmentation. Broadly speaking, graph cut clustering uses a graph representation of data in which nodes in the graph represent data points (generally pixels or superpixels in the case of image segmentation), and the weighted edges between nodes represent affinity between the data points. The algorithm produces from this initial graph two or more disjoint clusters of data points by cutting edges. The edges to be cut are chosen to optimize an objective function based on their weights, possibly under other problem-specific hard or soft constraints. Different objective functions and constraints produce different algorithms.

Methods based on segmenting an image into foreground and background regions can be augmented with a shape prior for the foreground region, implemented by adding a shape term to the objective function. One interesting example of this is the star shape prior proposed by Veksler [126]. A region is considered to be star-shaped with respect to a centre c if and only if for each point p on the border of the region, every point q on a line between c and p is inside the region. The centre is not necessarily unique, and often many choices of c will satisfy these properties. It is clear that any convex shape is also a star shape. Many other common shapes are as well; a star, a heart shape, and a chevron shape can all be star-shaped even though they are not convex. For interactive segmentation, the centre $c \in V$ of the foreground region is specified by the user.

The star shape prior can be implemented using pairwise shape constraints. For each pair of pixels $i, j \in V$ such that j lies on a line between i and c, the cost function

$$s_{ij} = \begin{cases} 0 & i, j \in R_{FG} \vee i, j \in R_{BG} \\ \infty & i \in R_{FG} \wedge j \in R_{BG} \\ \beta & i \in R_{BG} \wedge j \in R_{FG} \end{cases} \tag{C.9}$$

where R_{FG} and R_{BG} represent the foreground and background sets, respectively. Note that the cost of violating the star shape prior (the second line of the equation) is infinite; as a result, this imposes a hard constraint that the foreground region be a star shape. The constant β is used to counteract the bias of minimum cut segmentation algorithms toward short boundaries and therefore small regions; in the experiments described by Veksler, it is set for each instance to the minimum required to result in a region of at least a minimum size. Using a shape term consisting of the sum over all s_{ij}, the minimum cut solution can be found efficiently [126]. The strong results demonstrate the utility of a shape prior in segmenting natural images. Our page segmentation algorithm uses a shape prior in which regions are required to be rectangular; in our case, the use of the shape prior reflects the known properties of the domain and allows easier optimization of our probabilistic objective function.

Veksler's paper [126] has an additional interesting feature relevant to this review. The similarity measure adapts to the local distribution of pixel intensities in a window around the pixel pair. More specifically, the parameter σ in the equation $w_{ij} = \exp(-\frac{(I_i - I_j)^2}{2\sigma^2})$, which defines a similarity-based pairwise affinity between pixels, is set to the mean of the absolute intensity differences in a 20×20 pixel neighbourhood around the pixels i and j. This allows the sensitivity of the affinity to edges to adapt to local changes in the prevalence of edges. Although implemented differently, the objective is similar to our use of the local distributions of edge detector responses to estimate the probability that a possible edge is locally significant (see Sections 3.2.2 and 3.3.1).

C.2.3 Random Fields

One common method for Bayesian image segmentation is the use of Markov random field (MRF) models, which express spatially localized probabilistic relationships between regions. These models are defined for an undirected graph $G = (S, E)$, where E represents a set of edges and S represents a set of nodes or sites. Let X represent a set of random variables X_i, where X_i represents the label or value of the site s_i. When writing probabilities, we abbreviate the event "$X_i = x_i$" as simply "x_i" when this does not result in ambiguity, in the interest of brevity. The set of labels is denoted L. The graph represents the statistical dependence relationships between nodes. This model is a Markov random field provided that the following conditions hold:

- $\Pr(x_i|\{x_j|i \neq j\}) = \Pr(x_i|\{x_j|(s_i, s_j) \in E\})$
- $\Pr(x_i) > 0 \ \forall x_i \in L$

In the context of computer vision, sites are generally pixels or superpixels. For a pair of sites s_i and s_j, there is an edge $(s_i, s_j) \in E$ if and only if s_i and s_j are adjacent according to some neighbourhood system. In the specific problem of image segmentation, the objective is generally to determine which sites have the same label and therefore belong to the same cluster.

The Hammersley-Clifford theorem [55] shows that there is an equivalent energy-based formulation of the MRF as a Gibbs distribution. Let $\mathbf{x}$ represent an assignment of labels to all $X_i \in X$. Then

$$U(\mathbf{s}) = \sum_{c \in C} V_c(\mathbf{s}) \tag{C.10}$$

where C is the set of all ordered cliques in G and V_c represents the potential function for the clique c. The probability can be defined in terms of this energy as follows:

$$P(\mathbf{s}) = \frac{1}{Z} \exp\left(\frac{-1}{T} U(\mathbf{s})\right) \tag{C.11}$$

where T represents a temperature (assumed to be 1 unless specified otherwise) which controls the relationship between energy and probability (higher-energy configurations are less probable at low temperatures), and Z represents the partition function, acts as a normalizing constant. In practice, it is usually necessary to use an approximation of Z due to the difficulty of computing it exactly for models of a useful size (see, *e.g.*, [130]).

It is often simpler to use the Gibbs formulation of the problem in computer vision. Useful constraints in this domain, such as "adjacent sites tend to have the same label" are easily specified as potential functions on cliques. These potential functions may also depend on image data. The Hammersley-Clifford theorem is, therefore, very useful in the application of MRF models to computer vision problems.

D'Elia *et al.* proposed the use of an MRF model embedded in a binary tree for segmenting images with an unknown number of regions [39]. This model uses a recursive top-down approach to segment regions in a principled Bayesian framework. Each node t in the tree T contains a set $S^t \subseteq S$ of nodes, representing a region. Each node has a corresponding MRF X^t defined over S^t, a set of parameters θ^t to specify the the potentials of the corresponding Gibbs distribution, and potential functions V^t. If the node t has children, these are denoted $l(t)$ and $r(t)$ [39]. Let $p(t)$ represent the parent of t (for the root node, $p(t) = \emptyset$). The definitions given in [39] imply that child nodes cover their parent ($S^{l(t)} \cup S^{r(t)} = S^t$), and are disjoint ($S^{l(t)} \cap S^{r(t)} = \emptyset$); this is similar to our tiling constraints in the problem of web page segmentation, although the framework described by D'Elia *et al.* does not require that the regions be axis-parallel, rectangular, or even connected. For each node $t \in T$, L^t is a set of two labels, one for regions in $l(t)$ and one for regions in $r(t)$.

With these definitions in mind, a binary segmentation tree can be found be recursively optimizing (with respect to MAP) over $\mathbf{Pr}(\mathbf{x}^t|\mathbf{x}^{p(t)})$. The stopping condition is based on comparing two trees as hypotheses.

Related to the MRF model is the conditional random field (CRF) model [84]. The objective of the CRF model is to explicitly model the joint *conditional* probability distribution of a set of labels given a set of observations. In the case of image segmentation, the labels are generally region assignments and the image data constitutes the observations. The Hammersley-Clifford theorem [55] applies to CRF models [84] as well as to MRF models. In a CRF,

$$\Pr(\mathbf{s}|\mathbf{d}) \propto \exp\left(\sum_{c\in C} V_c(\mathbf{s})\right) \tag{C.12}$$

using the same notation as was used above to define an MRF. Note that, unlike in Equation C.11, the left hand side of the equation is a conditional probability.

In 2011, Krähenbühl and Koltun showed that efficient inference is possible in a fully connected CRF with Gaussian pairwise potentials for all pairs of pixels in the image [80]. Prior to this work, CRFs were limited to pixel-level local connectivity or superpixel-level full connectivity due to the computational complexity of optimizing over a number of pairwise edge potentials that was quadratic in the number of sites. Krähenbühl and Koltun demonstrated a method for using Gaussian filtering to perform the message-passing step in optimizing a mean-field approximation to the true underlying CRF distribution. As our own work does not use a CRF-based model, the details of this efficient optimization algorithm need not be discussed here; it is important, however, to note at least the existence of this algorithm in any discussion of modern CRF models in computer vision. In 2015, Zheng *et al.* demonstrated that the CRF model can be incorporated into recurrent neural networks to produce a hybrid structure, taking advantage of the capabilities of modern deep learning alongside probabilistic graphical models [140]. The method described by Zheng *et al.* implements steps of Krähenbühl and Koltun's filter-based mean-field iteration algorithm as CNN layers. This CNN layer formulation can be used in a recurrent neural network formulation.

Unlike MRF models, CRF models are not generative; a CRF (in the domain of image segmentation or semantic segmentation) is designed specifically to predict labels given image data, while an MRF model must learn the joint distribution over images and labels. This gives CRF models an advantage in the learning process; the smaller scope of learning in a CRF reduces the quantity of training data required [63].

Although MRF and CRF techniques are principled, Bayesian algorithms, they differ significantly from the approach taken in our algorithms. Our approach explicitly finds the

probability that specific cues truly exist in specific places in the image (*e.g.*, estimating the probability that an edge is significant with respect to its local neighbourhood, or that a semantically significant line exists between two points). The problem is divided into a series of stages, with higher-level cues building upon lower-level cues to produce the segmentation. Most segmentation algorithms based on random fields estimate the probability of an assignment of segmentation labels to the pixels (or superpixels) in the image. Our approach of more gradual assembly fits well with the edge-based approach (although edge-based potentials can certainly be implemented in random field models) and with the intended prior probability distribution over segmentation trees, and is an intuitively appealing approach to web page segmentation.

C.3 Region Classification

Region classification, in the sense used in this thesis, is the process of associating semantic labels with regions in a segmentation of an image. This approach is not as common in computer vision as simultaneous segmentation and classification (described in Section 7.1), or object detection. Object detection finds objects of a specific class or classes in an image, but only provides a coarse localization through a bounding box. Although not the largest area of research, there are computer vision algorithms that are designed specifically to perform region classification. It is also worth noting that region classification can be addressed using general classification methods from machine learning, with only the features used to describe regions being derived from computer vision.

One area in which region classification is especially popular is the analysis of three-dimensional data. Zhu *et al.* described [141] a segmentation-classification pipeline for range images, where the value at a given pixel is a function of the distance from the sensor to the nearest object along the corresponding line of sight, rather than of the light hitting the corresponding pixel in the camera sensor. In this pipeline, the depth image is segmented using a graph-based technique, and the regions are classified by a support vector machine (SVM), based on statistical features of the set of points in space corresponding to the region. Sérna and Macrotegui described [116] a pipeline for elevation images formed by projecting a three-dimensional point cloud to a horizontal plane. This pipeline begins with the separation of the ground, followed by segmentation of connected object using a morphological method, and classification using geometrical features, the number of adjacent objects, and colour features if they are available. Sérna and Macrotequi also provide a good overview of similar algorithms in this domain.

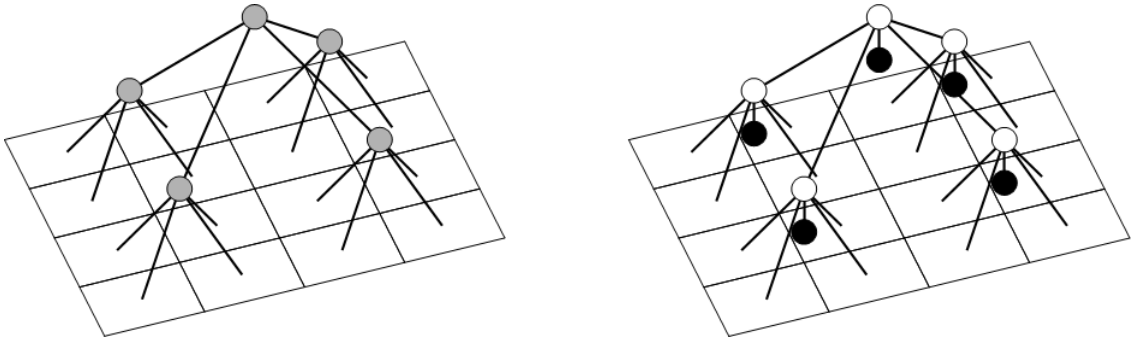
The use of separate segmentation and classification steps for point clouds, depth images,

and elevation images is a good example of selecting a vision pipeline to suit a domain. In these images, the depth information provides rich three-dimensional information about object contiguity which can be used in the segmentation process. Similarly, web page images offer important division cues such as whitespace borders, background colour transitions, and aligned edges that provide evidence for divisions between objects in the absence of specific knowledge about the labels of regions.

Although common in the domain of three-dimensional data, separate classification of a segmented image is not restricted to this setting. An interesting method proposed by Fredembach *et al.* [49] takes a particularly interesting approach. In most cases, the image axes and borders in a natural image are assumed to be arbitrary. The approach taken in [49], however, takes advantage of the expected patterns of composition in photographs—the sky, for example, can be expected to be in the upper part of the image. The "eigenregion" approach combined shape and position modelling, and with fairly broad classes for regions was found to have advantages for image classification.

Our region classification approach represents a novel classification method well-suited to the problem of classifying regions embedded in a hierarchical segmentation tree. The use of an HMT with a global structure corresponding to the global structure of the segmentation tree provides a sophisticated and principled means of accounting for context without requiring context features. It allows joint inference of the maximum *a posteriori* assignment of labels over all regions in the image, while the guaranteed tree structure of the model allows efficient inference. By maintaining consistent local structures, we ensure that a single set of conditional probability distributions—which can be readily learned from labelled data—is sufficient to define the model for any segmentation tree despite the resulting changes in the global structure of the HMT. Although designed for the domain of web pages, we believe that our algorithm has potential for generalization to other domains; this is an area that we hope to investigate further in the future (see Section 8.3.3). Because of its principled design and flexibility in representing contextual information, our region classification algorithm represents an important contribution in computer vision more generally.

While our HMT-based approach to region classification is novel, hidden Markov trees have been used in other ways in computer vision, especially when working in the wavelet domain. Although wavelets are a large and interesting area of computer vision, our discussion will be restricted to a high-level description of the key features for contrasting our use of the HMT model with its use in the wavelet domain. Wavelets are a class of alternative representations of images; the wavelet transform is broadly analogous to the Fourier transform in that it is a change of basis to an alternative representation which is more appropriate for some types of analysis. The spatial-domain representation of the



(a) Diagram of wavelet scales, showing quadtree structure. Grey circles represent wavelets, with lines connecting the parent wavelets to their children.

(b) Diagram of an HMT corresponding to the wavelets shown at the left. White circles represent hidden states, and black circles represent wavelet coefficients.

Figure C.1: Diagrams of wavelet structure and a corresponding HMT, based on diagrams from [111]. Each diagram represents a single subband.

original image uses basis functions (single-pixel impulses) which are localized in space but not in frequency, and the frequency-domain representation provided by the Fourier transform uses basis functions which are localized in frequency but not in space; the wavelet domain uses basis functions (the wavelets) which are localized in frequency and in space. The wavelet domain uses repetition of the same functions at multiple scales to form an orthonormal basis for the space of greyscale images; large wavelets cover the same area as a 2×2 grid of child wavelets. Figure C.1a shows this hierarchy. Many functions can be used as the basis of a wavelet transform.

The quadtree structure of the wavelet domain makes a tree-structured probabilistic graphical model a natural approach to the problem. Additional features of the wavelet domain make an HMT structure especially attractive. Romberg *et al.* described [111] an HMT model based on the statistical properties of the wavelet domain in natural images. Wavelet coefficients have been observed to be large at edges and small elsewhere; this not only makes them potentially useful as edge detectors, it also means that they are produced by a mixture of an edge-containing distribution which favours large coefficients and a non-edge-containing distribution which favours small coefficients. In an HMT model of natural images, it is therefore reasonable to provide a hidden state which indicates whether the corresponding coefficient is expected to be large or small. Another observed feature of the properties of wavelets in natural images is that coefficient magnitude tends to persist across

scales (*i.e.*, the child wavelets of a parent wavelet with a large coefficient also tend to have large coefficients). It is, therefore, reasonable for the hidden state of a parent to influence the hidden states of its children. Figure C.1b shows a schematic of an HMT of the form proposed by Romberg *et al.* In [111], the statistical model is explored in much greater depth and many aspects of the statistics of wavelets in natural images are examined; here, however, it is only necessary to discuss the most important aspects for determining the structure of the HMT.

While the structure of the HMT shown in Figure C.1b appears superficially similar to the structure of our own segmentation tree, there are important differences. Aside from its size, the structure of the wavelet HMT is fixed, not only locally but globally. It is not based upon a segmentation which adapts to the structure of the image, but rather on fixed fields in the image. Rather than representing a high-level object or region class, the hidden states in the wavelet HMT represent components of a mixture model from which the wavelet coefficients are drawn. We do not claim that the wavelet HMT is inferior to our own HMT classification model; it is a powerful model of the statistical structure of an image with many applications. Our HMT for classification simply solves a different problem, and so while it interesting to compare the two in terms of structure, the models do not compete with each other.

In [110], Romberg *et al.* used an HMT to classify textures in an image. The system simultaneously segments the page into regions of consistent texture and classifies the textures. This classification algorithm works at the level of textures and uses a fixed tree structure. Our classification algorithm uses much higher-level classes, and adapts its structure to an existing segmentation tree. Despite the superficial similarities, our system is very different in its assumptions and its approach to the classification problem.

Appendix D

Example Screen Reader Transcript

In this section we present an example transcript from the NVDA 2018.2 screen reader program, as a sighted novice user browses the Weather Network forecast page for the town of Sackville, New Brunswick. This page was collected on July 8, 2018, and the transcript represents an ultimately successful but time-consuming attempt to find the seven-day forecast. For the purposes of this example, default settings were used, including for verbosity settings. The original page is shown in Figure D.1, with the seven-day forecast (where the transcript stops) circled. Note in the transcript that a great many structural features are read out, including heading levels and the presence of links. Browsing features used included straightforward linear browsing and jumping to the next (or previous) occurrence of a specified type of element, such as a heading, a link, or an image.

```
Sackville, New Brunswick 7 Day Weather Forecast - The Weather Network  document
Your weather when it really mattersTM
Sackville, New Brunswick Weather
heading  level 1
SHORT TERM FORECAST
heading  level 4
WATCH TRENDING VIDEOS
heading  level 4
SHORT TERM FORECAST
heading  level 4
link
This Afternoon 12pm  6pm 25C
link
```

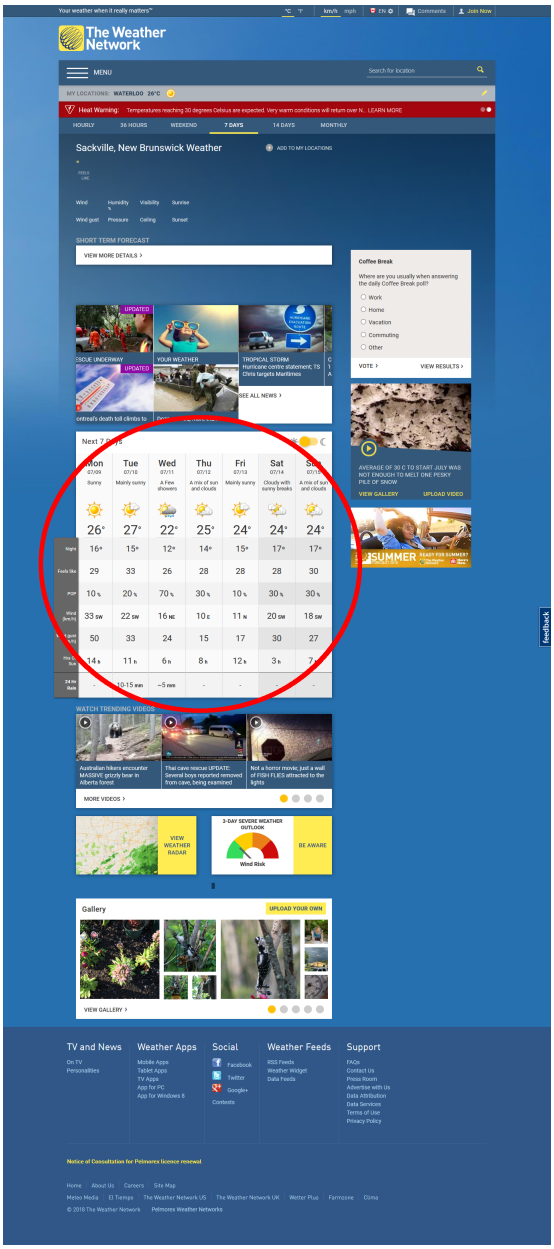


Figure D.1: Original page, showing the circled seven-day forecast.

Sunny in the afternoon.
link
Tonight 6pm 6am 15C
link
Mainly sunny in the evening remaining clear overnight.
link
VIEW MORE DETAILS
link
UPDATED
link
RESCUE UNDERWAY

link
Several boys rescued from Thai cave; attempt resumes Monday
graphic
Several boys rescued from Thai cave; attempt resumes Monday

link
YOUR WEATHER

link
Heat warnings resume for Maritimes as humidity returns
graphic
Heat warnings resume for Maritimes as humidity returns

WATCH TRENDING VIDEOS
heading level 4
3-DAY SEVERE WEATHER OUTLOOK
heading level 2
Gallery
heading level 2
Coffee Break
heading level 4
 content info landmark
TV and News
heading level 3

```
Weather Apps
heading  level 3
Social
heading  level 3
Weather Feeds
heading  level 3
Support
heading  level 3
no next heading
no next heading
no next heading
Weather Feeds
heading  level 3
Social
heading  level 3
Weather Apps
heading  level 3
TV and News
heading  level 3
Coffee Break
heading  level 4
Gallery
heading  level 2
3-DAY SEVERE WEATHER OUTLOOK
heading  level 2
WATCH TRENDING VIDEOS
heading  level 4
SHORT TERM FORECAST
heading  level 4
Sackville, New Brunswick Weather
heading  level 1
no previous heading
no next table
list with 5 items

Work  radio button  not checked

clickable
```

Work
 content info landmark
list with 2 items
On TV
link
list with 5 items
Mobile Apps
link
Weather Apps
heading level 3
TV and News
heading level 3
Coffee Break
heading level 4
Gallery
heading level 2
3-DAY SEVERE WEATHER OUTLOOK
heading level 2
WATCH TRENDING VIDEOS
heading level 4
SHORT TERM FORECAST
heading level 4
Sackville, New Brunswick Weather
heading level 1
A few clouds
graphic
Several boys rescued from Thai cave; attempt resumes Monday
graphic
Heat warnings resume for Maritimes as humidity returns
graphic
Hurricane centre statement; TS Chris targets Maritimes
graphic
'I should have been dead,' Albertans fend off 3 grizzlies
graphic
Storm threat, sweltering temperatures ahead for Prairies
graphic
Canadian firefighter dead after falling from K2 mountain
graphic

Montreal’s death toll climbs to 33, heat warnings end
graphic
Dozens dead, more than a million evacuated in Japan flooding
graphic
Sunny
graphic
Sunny
07/09
Mon
24 Hr Rain
Hrs Of Sun
Wind gust (km/h)
Wind (km/h)
POP
Feels like
Night

clickable
Show/Hide

Next 7 Days
link
SEE ALL NEWS
Next 7 Days

clickable
Show/Hide

Night
Feels like
POP
Wind (km/h)
Wind gust (km/h)
Hrs Of Sun
24 Hr Rain
Mon
07/09
Sunny

```
graphic
Sunny
26

16

blank
29
10%
33SW
50
14h
```