Vol.2009-HCI-133 No.8
2009/5/16

ScoutView：Webページにおけるナビゲーション支援インタフェース

井桁 正人[†1]　寺田 実[†1]　丸山 一貴[†2]

長大なWebページを表示する際に，ページ全体のサムネイルを用いるのは一般的な手法である．しかし，この手法では表示が縮小されるため，閲覧したい場所まで移動しなければその詳細は得られない．

そこで，サムネイルにルーペ機能を付加し，移動前に詳細を確認可能とした．また，ページ内検索の結果を視覚的なスニペットの一覧提示し，検索キーワードのコンテキストを把握可能とした．このようなインタフェースを試作，評価した．

ScoutView：Scrolling support interface for Web page

Masato Igeta,[†1] Minoru Terada [†1]
and Kazutaka Maruyama[†2]

To navigate in a very long Web page, the thumbnail of the entire page is helpful. But the user have to move to the location in order to check the detail of the thumbnail.

We added a magnifying glass to the thumbnail, making it possible to check the detail without changing the current location. We also overlay a list of visual snippets of the search results on the page, which offers the visual context of the search keyword. Results of user experiments are also included.

†1 電気通信大学電気通信学研究科情報通信工学専攻
Graduate School of Information and Communication Engineering, The University of Electro-Communications
†2 東京大学 情報基盤センター
Information Technology Center, The University of Tokyo

1. はじめに

近年，通信サービスの高度化やWiki・ウェブログなどのWebページ作成支援サービスの充実により，Webページの数が増加している．それに伴い，情報量が豊富で長大なWebページも数多く存在している．

そこで，その中から所望の情報を見つけ出すには，Google[1)]などの検索エンジンを利用することが有効である．しかし，ユーザが所望の情報を得るためには，Webページを検索エンジンにより見つけ出すだけでは不十分である．

検索したWebページには，一つ一つの長さが一画面を超えてしまう場合が多数存在し，Boungwon Suhら [2] によると，69% のページが一画面以上の長さを持つことが分かっている．また，長大なページの場合には複数の話題が存在すると推測できる．そのため，引き続きページの閲覧の際にも，その中から所望の情報を探索する必要がある．

一般的なWebブラウザでは，ユーザが見ているページ (以下，閲覧ページとする) の内，表示されるのは一画面分だけである．そのため，閲覧ページの長さが一画面を超え，所望の情報が表示画面内に存在しない場合には，所望の情報の存在する位置の探索 (以下，ページ内ナビゲーションとする) をしながら，現在表示されている位置を移動 (以下，スクロールとする) する必要がある．

そのようなページ内ナビゲーションの方法は，以下で述べる概観からのナビゲーションとキーワードからのナビゲーションの大きく2つに分けられる．

1.1 概観からのナビゲーション

概観からのナビゲーションとは，ページをスクロールする際の一般的な手法であり，マウスホイールやキーボードのショートカットキーにより一定量のスクロールを繰り返すことで，所望の情報を探索することである．しかし，スクロールを実行する前にスクロール先の内容が見えないので，どこまでスクロールすればよいか判断することができない．その結果，少量のスクロールを繰り返すこととなり，閲覧ページが長大な場合には，ユーザへ大きな負担となってしまうという問題がある．

1.2 キーワードからのナビゲーション

キーワードからのナビゲーションとは，ブラウザに搭載されているテキスト検索機能を用い，キーワードを指定することで，所望の情報を探索することである．しかし，キーワードがページ内の複数の位置に存在する場合，その位置ごとに表示画面の急激な移り変わりが生じる．これを繰り返すたびに表示内容を把握し直さなくてはならないため，ユーザへの負担

Vol.2009-HCI-133 No.8
2009/5/16

が大きくなってしまうという問題点がある．

2. 関連研究

文章に対する閲覧支援として，重要であろうと思われる部分をハイライトするなどして強調することは一般的な手法である．この手法は検索エンジンからのクエリをハイライトするGoogle ツールバー[2] でも用いられている．

この手法に対して，前述のような一画面でページ全体を表示できないという問題点を改善するために，ページの通常のブラウザ表示に加えて，ページ全体のサムネイルがよく用いられる．そのような関連研究として，PopoutPrism[5]，StackOverview[6] が挙げられる．

PopoutPrism は，Web ページの文章に対する閲覧支援を目的とするシステムである．Web ページに対するサムネイルと詳細の二つの提示方法においてその両者でキーワードの位置を強調して提示する (図 1)．また，サムネイル上でダブルクリックを行うと，対応する位置までのスクロールを行うことが出来る．これにより，閲覧ページ内におけるキーワード位置が把握でき，その位置まで即座にスクロールを行うことができる．

StackOverview は，Web ページへの再訪問時における情報収集支援を目的とするシステムである．PopoutPrism と同様にサムネイルと詳細の二つを提示提示方法を用いている (図 2)．しかし，StackOverview ではユーザが訪問している Web ページ内の情報を蓄積し，再訪問時に更新差分 (追加情報，削除情報，ユーザ指定の注目情報) の強調を行う．これにより，閲覧ページ内における前回の訪問から更新された記事の位置を把握でき，その位置まで即座にスクロールを行うことができる．

この様にサムネイルによりページの概観をユーザへ提示することで，

- 強調された部分へ即座にスクロールできる
- 強調されていない，即ち重要ではないと思われる部分へのスクロールを省ける

という利点が生じると考えられる．

また，表示ページの上下両端を魚眼レンズのように縮小するものとして，Fishnet[7] が挙げられる．Fishnet は，PopoutPrism と同様に Web ページの文章に対する閲覧支援を目的とするシステムである．閲覧ページに対して，その上下を魚眼レンズのように縮小する提示方法において，キーワードの位置を強調している (図 3)．これによって，ページ全体を一画面で表示可能となり，サムネイル上のクリックにより，対応する位置までスクロール可能となる．

この手法の利点は，サムネイルを提示する場合と比較して，サムネイルと詳細の視点の切り替えを行う必要がなく，また画面上のスペースも節約することが出来るという点である．

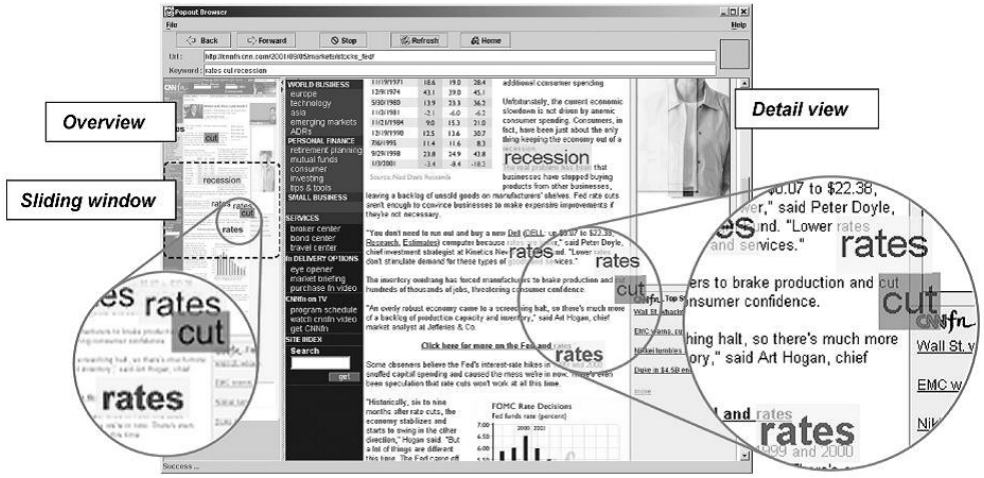

図 1 PopoutPrism[5]
「cut，rates，recession」というユーザが指定したキーワードが強調されている．

しかし，これらのシステムにおいて，ページのサムネイルだけではスクロールする前に強調された箇所の詳細な内容までは判断することができず，注目すべき場所の候補を挙げただけであると言える．そのため，真にユーザが閲覧すべき場所であるかどうかを特定するためには，その場所までスクロールしなければならないことが問題となる．また，用いられている閲覧ページのサムネイルは，単に閲覧ページを画面の高さに縮小しているのみであり，より長大なページの場合には，縮小率が上がり，サムネイルの描画内容が読みづらく，閲覧ページの概観を判断できないものとなってしまう．

これらのシステムはあくまでページを「読む」ための支援の一つであった．しかし，情報探索を行う際に長大なページに遭遇した場合には，その全体を読む必要はなく，所望の情報について書かれた部分だけ閲覧すれば十分であると言える．よって，そのような場合には，ページについて「読む」支援とは別にページ内を「探す」支援をする必要がある．

このような問題点を解決するために，提案システム ScoutView について試作，評価を行った．

Vol.2009-HCI-133 No.8
2009/5/16

図 2 StackOverview[6)]
前回の訪問から追加された更新箇所が赤色で強調されている．

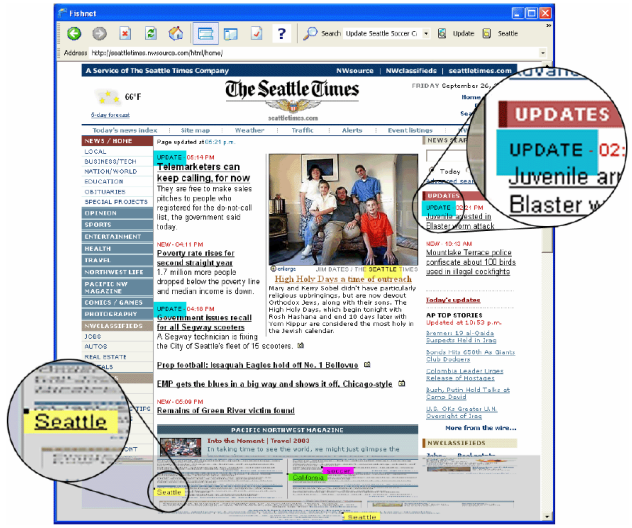

図 3 Fishnet[7)]
魚眼レンズのようにページの下端が縮小されている．

3. 提案システム：ScoutView

3.1 概　　要

本研究では，前章で述べた問題点を解決し，ページ内ナビゲーションを支援するために，

- 閲覧ページの概観やキーワードによる注目箇所の探索を支援
- 注目箇所の詳細を提示
- 注目箇所までのスクロールを支援

という要素を備えたインタフェースを提案する．また，注目箇所の詳細の提示方法については，概観からのナビゲーションとキーワードからのナビゲーションそれぞれに沿った支援を行う．

ここで，注目箇所とは概観もしくはキーワードから得られた所望の情報が存在すると思われる箇所のことである．よって，注目箇所に所望の情報が必ず存在するとは限らず，その詳細から真にユーザが閲覧すべき場所であるかを確認する必要ある．

これにより，関連研究における「サムネイルから閲覧ページのレイアウトを把握することは出来たが，その部分の詳細を把握することは出来ない」という問題を解決可能となる．その結果，閲覧するべきかどうかの判定をするためだけにスクロールをする必要がなくなり，閲覧ページ内でよりスムーズに所望の情報を見つけることができる．あとは，そのスクロール先の文章より，所望の情報を得ることが可能となる．

3.2 機　　能

3.2.1 全体サムネイルとルーペ窓

概観からのナビゲーションを支援するために，関連研究と同様にブラウザのサイドバーに閲覧ページ全体のスクリーンショット (以下，全体サムネイルとする) を提示する (図 4)．この全体サムネイルはクリックすると対応した位置までスクロールを行う．これによって，ページのレイアウトを把握できるので，読みたい記事がある位置を予測する事ができる．また，スクロールしなければ見えない位置にある情報についても把握することが可能となる．

縮小されたスクリーンショットが一画面を超えてしまう場合には，全体サムネイルの分割を行うことにより，全体サムネイルの描画が崩れるのを防いでいる．分割されたサムネイルはマウスのクリックかホイールで，その分割ページ数を切り替えることができる．

さらに，この全体サムネイルに対して，その上にマウスカーソルを載せると，その近傍を

Vol.2009-HCI-133 No.8
2009/5/16

部分的だが縮小されていない原寸大の閲覧ページのスクリーンショット (以下，ルーペ窓とする) をページの表示上に提示するようにした (図 4)．全体サムネイルは閲覧ページの概観を得るには有効だが，縮小されてしまっているため，細かい文字や画像などを判別することができず，注目箇所が真に読みたい記事であるかを判断することはできない．，ユーザがサムネイル上で注目したいと思った場所にマウスカーソルを合わせれば，その箇所の詳細について把握することができるようになる．これによって，ユーザはスクロールすることなしに，注目しようと思った部分の詳細を把握することが可能となり，真に閲覧すべき場所であるか判断することができる．

このように，(1) 全体サムネイルでページのおおまかな内容を把握する，(2) 所望の情報がありそうな注目箇所の詳細をルーペ窓から得る，(3) 閲覧すべきだと判断したら全体サムネイルの該当箇所をクリックしてスクロール，という手順により，所望の情報の存在する位置までスクロールを行うことが可能となる．

関連研究においてはサムネイルから閲覧ページのレイアウトを把握することは出来たが，その部分の詳細を把握することは出来なかった．しかし，本システムではサムネイルにルーペ窓を加えることで，スクロール先の情報を前もって知ることができる．これによって，閲覧するべきかどうかの判定をするためだけにスクロールをする必要がなくなり，概観からのナビゲーションにおいて，閲覧ページ内でよりスムーズに所望の情報を見つけることが可能となる．

3.2.2 近 傍 窓

キーワードからのナビゲーションを支援するために，キーワードによるテキスト検索を行う．そのために，キーワード近傍のスクリーンショット (以下，近傍窓とする) の一覧を提示する (図 5)．

この近傍窓はクリックすることで，対応する箇所へのスクロールを行う．テキスト検索で複数の箇所が見つかった場合に，所望のものとは全く異なったコンテキストでキーワードが使われている箇所に対してまでスクロールすると，それは無駄な手間となってしまう．

そこで，ユーザへ近傍窓を提示することにより，テキスト検索によって見つかった箇所のキーワードのコンテキストを把握可能となるので，その無駄な手間を省くことが可能となる．

ここで言うキーワードのコンテキストとは，文章中の言葉の意味合いだけでなく，HTML 要素の種別も含まれている．例としては，そのキーワードが，

- 項目のメニューやタイトルのため，フォント (文字のサイズや色) が大きくなっている
- 別の記事へのリンク

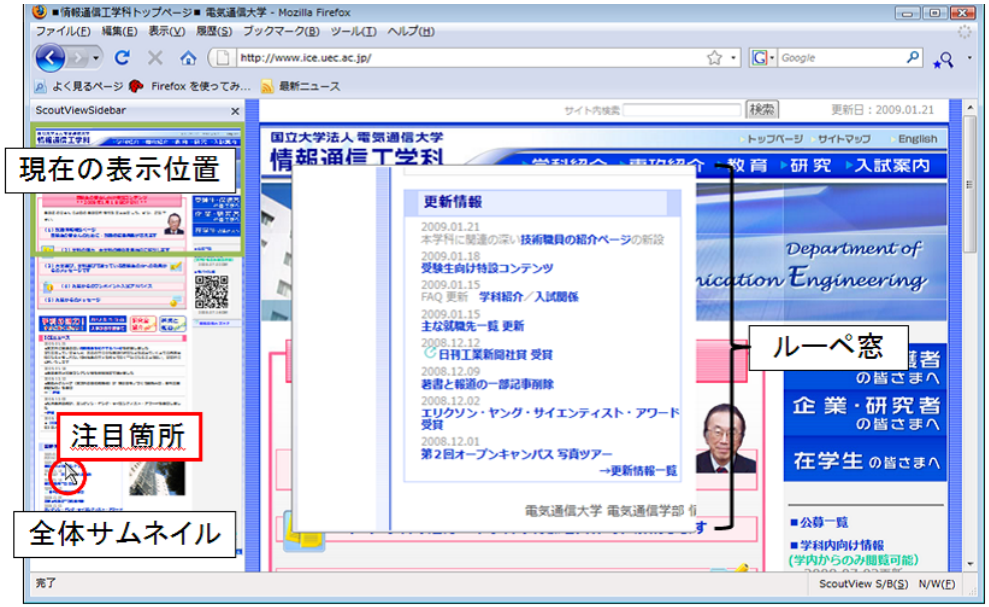

図 4 全体サムネイルとルーペ窓

- 表の一部
- 図のキャプション

の場合について，近傍窓により把握することが可能となる．

また，閲覧ページ内において，キーワードはいくつかの集団にある程度密集していることが予測できる．そこで，複数のキーワードが隣接している場合には，一つの近傍窓に収まるようにし，ユーザが近傍窓同士の近さを把握できるように，近くに存在する近傍窓を線 (以下，近傍線とする) で結ぶようにした．

関連研究の PopoutPrism では，キーワードの位置を把握できるようになるだけであった．そのため，複数の位置にキーワードが存在した場合には，どの位置を閲覧すればよい判断ができない．しかし，本システムでは近傍窓により，位置を理解させるのではなく，キーワードのコンテキストを把握させることで，閲覧箇所の選別を容易に行えるようにした．

3.3 実 装

ScoutView は，Web ブラウザ Firefox[3] の拡張機能として実装を行った．これは，ブラ

Vol.2009-HCI-133 No.8
2009/5/16

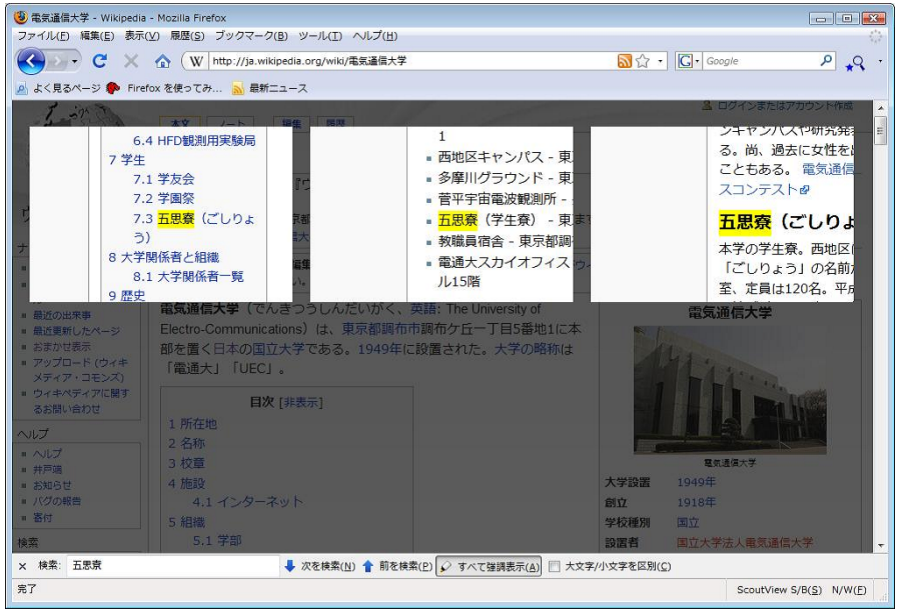

図 5 近傍窓
キーワードを「五思寮」として近傍窓を実行した場合．このとき,「五思寮」についての解説が所望の情報であれば，3 つの近傍窓の内の右側にあるものを選択すればよい．

ウザ自体の入手や，拡張機能の配布が容易になるという利点があるためである．

また，スクリーンショットの描画については，HTML の描画要素である canvas 要素[4] を用いて行った．

4. 評価実験

提案システムの評価について述べる．評価は，著者が実験用に作成した実験ページに対して 2 つのページ内ナビゲーションスタイルに従って，目標となる情報が存在する位置までのスクロールにかかる時間の測定と，総合的な使いやすさをアンケートにより評価した．

また，被験者は本学の学部 4 年生，修士 1，2 年生の計 8 名である．全員，習慣的に Web ブラウジング行っている．本実験では，解像度 1920 × 1200 を持つディスプレイと Core2 Extreme 3.0GHz とメモリ 2GB を搭載した PC 上でタスクを行った．

4.1 タスク

以下ので述べる各タスクについて，システムの不使用・使用の二つの条件で，システム不使用の場合は従来システム (マウスホイールによるスクロールとキーワード検索) を，システム使用の場合は ScoutView を用いてタスクを実行させた．また，システムの不使用・使用の順序によって実験結果に差が生じないように，その順番を被験者の半数には不使用条件から，もう半数には使用条件から実行するようにした．

実験ページは以下の図 6 のようなレイアウトとなっている．両タスクともウインドウサイズを 1000 × 1000 ピクセルとした．なお，タスク 1 において，サイドバーが必要な場合には，サイドバー幅を 200 ピクセルとしてウインドウサイズを 1200 × 1000 ピクセルとした．

タスク 1 で用いた実験ページ

実験ページ 1 の高さは 9000 ピクセル程度であり，1000 × 1000 ピクセルのウインドウサイズの場合，11 画面分程度の高さとなっている．すべて異なる 18 枚の画像が 3 列おきに等間隔に並んでいる．目標となる画像は，システムの不使用・使用の両条件でも最下行とその一つ上の行の 6 枚の内のいずれかとした．

タスク 2 で用いた実験ページ

実験ページ 2 の高さは 7000 ピクセル程度であり，1000 × 1000 ピクセルのウインドウサイズの場合，8 画面分程度の高さとなっている．テキストと画像が混在したページ上でキーワードを含む文字列が 9 個配置されている．目標となるコンテキストで用いられているキーワードの位置は，タスク 2(近) の場合は 2 または 3 個目，タスク 2(中) の場合は 5 または 6 個目，タスク 2(遠) の場合は 8 または 9 個目となっている．

4.1.1 タスク 1

概観からのナビゲーションを行った場合についての評価を行う．指定された画像を探しだすタスクを，システムを使用した場合と使用しなかった場合で 5 回ずつ繰り返し行った．

実験ページ 1 のような閲覧ページにおいて，タイトルとどのような画像であるかの説明を与え，その条件に見合う画像を探してもらった．閲覧し始めてから目標となる画像を見つけクリックするまでの時間を測定した．これにより，目標の候補となる部分とそうではない部分が混合している閲覧ページにおいて，概観からのナビゲーションによりページ内ナビゲーションを行った際にかかる時間を測定することができる．

具体例としては,「“オレンジ色の魚”というタイトルの魚の画像を探せ」というような説明を与えた．

時間の測定方法としては，実験ページ上部に設置されたストップウォッチにて行った．ペー

Vol.2009-HCI-133 No.8
2009/5/16

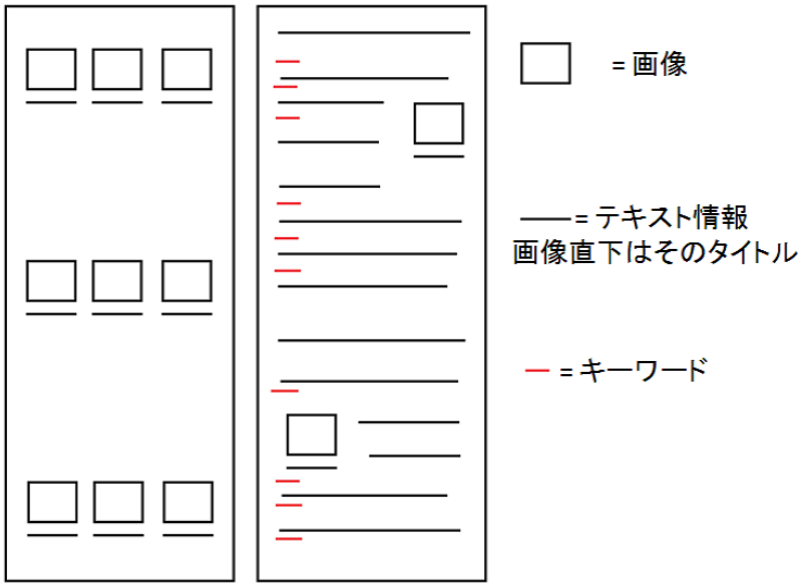

図 6 実験ページ 1(左) と実験ページ 2 のレイアウト (右)
実験ページ 1 では，画像以外の部分は空白であり，実験ページ 2 では文字が密に詰まっている．

ジ上部に設置されたボタンをクリックすることにより測定を開始する．そして，目標の画像をクリックするとそれまでにかかった時間をダイアログで表示するようにした．

なお，タスク 1 では目標となる画像をこちら側から指定した．タスク 1 においては目視による検索を比較するため，キーワード検索機能は行わないよう被験者へ説明した．

4.1.2 タ ス ク 2

キーワードからのナビゲーションを行った場合において，指定されたコンテキストで用いられているキーワードを探し出す．システムを使用した場合と使用しなかった場合を 15 回ずつ繰り返しタスクを行った．

実験ページ 2 のような閲覧ページにおいて，キーワードをこちらから前もって指定し，その前後の文字列がどうなっているかの説明 (コンテキスト) を与えることで，条件に見合うキーワードを探してもらった．そこで，キーワードが入力されている状態で閲覧し始めてから，目標となるキーワードを見つけ，クリックするまでの時間をシステムを使用した場合と使用しなかった場合について測定した．

これにより，複数の位置にキーワードが存在するような閲覧ページにおいて，キーワードからのナビゲーションを行う．そして，所望のコンテキストでキーワードが使われている場所までスクロールするのにかかる時間を測定することが出来る．

なお，タスク 2 ではキーワードとコンテキストのそれぞれを指定したが，コンテキストも含めたキーワード検索を行うことを禁じた．例えばキーワードが「aaa」で，コンテキストが「前後が*で囲まれているもの」であるとき,「*aaa*」を直接検索することを禁じ,「aaa」による検索だけを利用するようにした．また，その他のものは,「baaab」や「faaaf」のようなものとなっており，時間の測定方法はタスク 1 と同様である．

また，候補の一覧表示により，システムを使用しなかった場合とした場合で，目的の情報の出現位置によって，タスク達成の時間は異なることが予想される．即ち，最初の方の順番にある場合は使用しない場合のほうがより速く目標の情報を見つけだすことができるが，後ろの方の順番にある場合はシステムを利用した場合のほうがより速く，目標の情報を探し出すことができると予測できる．これを示すために，実験ページを目標の情報の出現位置が閲覧ページの上部，中央，下部の三段階で 3 種類用意した．各出現位置は 15 回の繰り返しの中に，5 回ずつランダムな順序でタスクを行った．

4.1.3 アンケート

タスク 1 とタスク 2 のタスク達成時間に加えて，アンケートを実施した．アンケートは全ユーザ 8 人を対象に，タスク 1 と 2 の終了後，被験者にアンケートを行った．各タスクでシステムの有無でどちらが使いやすかったか (1 がシステム無，4 がシステム有)，今後もこのシステムを使いたいか等の評価を 4 段階で行った．また，その他意見なども尋ねた．

4.2 実 験 結 果

4.2.1 タスク達成時間

タスク 1 とタスク 2 のタスクを達成するまでにかかった時間は図 7 のようになった．なお，図 7 におけるエラーバーは標準偏差により求めている．

4.2.2 アンケート結果

各タスクの使いやすさと継続利用については表 1 のような結果となった．各タスクの使いやすさについては，従来システムと ScoutView のどちらがタスクを実行し易かったかを四段階で尋ねた．

Vol.2009-HCI-133 No.8
2009/5/16

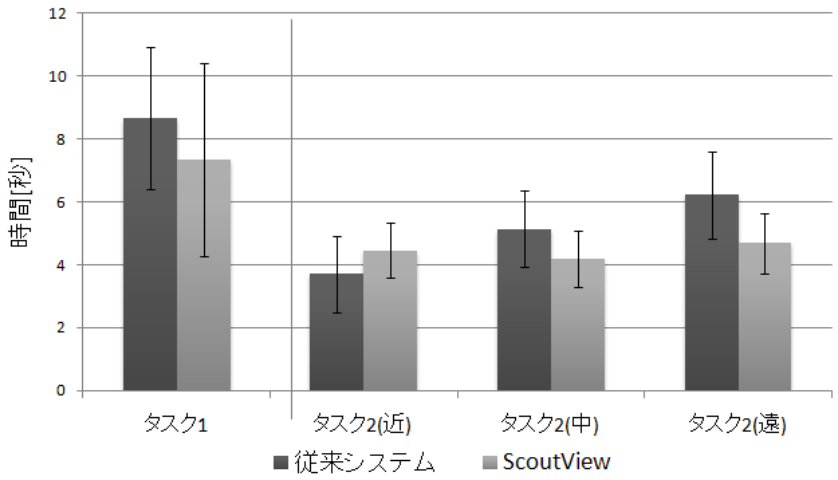

図 7 タスク達成時間

表 1 使いやすさと継続利用についての四段階評価

	タスク 1 の使いやすさ (1:従来システム,4:ScoutView)	タスク 2 の使いやすさ (1:従来システム,4:ScoutView)	継続利用 (1:使いたくない,4:使いたい)
平均	3.4	3.3	3.4

4.3 考　　察

4.3.1 概観からのナビゲーションに対する考察

タスク 1 について，図 7 より，個人による差が大きいため，統計的に有効性を検証する．どちらのタスクが達成に時間がかかっているかを明確にするために，帰無仮説を「システムの使用の有無による両群の母代表値に差がない.」，対立仮説を「システムの使用の有無による両群の母代表値に差がある.」として Wilcoxon の符号付き順位和検定を有意水準 5%の片側検定として行った．

その結果，タスク 1 においてシステムの不使用・使用状態で有意差が存在し，システムを使用した場合の方が有効であることが示せた．これにより，ページ内ナビゲーションによる時間を軽減することができ，ユーザへの負荷を軽減できたと言える．従って，概観からのナビゲーションにおいて，提案手法は有用であることが示せた．

アンケートでは，全体サムネイル上のクリックでのスクロールや，ページボックスにおけるマウスホイールのページ切り替えは好印象だった．これにより，関連研究より長大な閲覧ページにも対応でき，全体サムネイルとページボックスから，容易に概観からのナビゲーションを行うことができると言える．

しかし，ルーペ窓について否定的な意見も存在し,「ルーペ窓は視点を動かすのが手間である」という意見があった．これは，ルーペ窓の表示内容に注目すると全体サムネイル上のマウスポインタの位置がわからなくなってしまい，その表示内容がページ内のその部分の詳細であるかを判断できなくなってしまうためであると考えられる．

また,「ルーペ窓の大きさはもっと大きいほうが良い」という意見もあった．これについては，注目すべき部分の大きさは，大きければ良いというわけではなく，ユーザごとで一見して把握できる情報量には個人差がある．よって，小さすぎると狭いと感じるが，大きすぎると結局どの部分に注目すれば内容を一見して把握できなくなってしまうと考えられる．従って，ルーペ窓の改善点として，実装方法を変更することにより処理の重さを軽減し，ルーペ窓のサイズや表示・非表示をユーザが任意に選択できるようにすることが考えられる．

4.3.2 キーワードからのナビゲーションに対する考察

タスク 2 について，図 7 より，時間は個人による差が大きいため，タスク 1 の際と同様に Wilcoxon の符号付き順位和検定を有意水準 5%の片側検定として行った．

その結果，タスク 2(近) の場合は有意差が認められず，タスク 2(中) とタスク 2(遠) の場合に有意差があった．従って，タスク達成時間が目的の情報の出現位置によって異なるという予想通りの結果となった．

目標とするキーワードの出現が早い場合は，システムを使わない方が時間がかからないという結果となった．しかし，それ以外の場合には提案システムを利用することにより，そのタスク達成時間を短縮することができ，ユーザへの負荷を軽減できることが示せた．

これは，検索結果の一覧表示により，タスク達成時間が均一となったためであると考えられる．図 7 を見てわかるように，従来システムの場合にはターゲットの位置により時間が右肩上がりとなっているが，ScoutView の場合は時間が均一になっている．

アンケートの意見により,「キーワードの周りが見えて良かった」,「一覧なので，前の結果に戻らなくてよい」,「スクロールすると，キーワードが表示画面中央に来るのが良かった」と全体的に見て，好印象であった．2 番目と 3 番目の意見に対しては，従来の閲覧ページ内テキスト検索における,「ターゲットを見つけても，ボタンを連打しているために行き過ぎてしまう」,「キーワードが画面の下端，あるいは上端にくるようにスクロールされてしま

う」という問題点が改善されたためであると考えられる．

しかし，近傍窓について否定的な意見も存在し,「キーワードを入力したら，すぐに実行してほしい」,「文章のコンテキストはわからない」という意見があった．近傍窓を実行する際の改善点として，キーワードの入力完了後に一定時間過ぎると近傍窓を実行するようにして，実行負荷をより軽減する必要がある．また，文章のコンテキストについては，近傍窓で表示される範囲内にそのキーワードのコンテキストを類推できる単語などが含まれていない場合には，判断できないためであると考えられる．

4.3.3 ユーザビリティ

タスクの実行し易さについて，表 1 より，平均値はタスク 1 では 3.4，タスク 2 では 3.3 であった．これより，両タスクとも ScoutView の方がタスクを実行しやすかったことがわかる．従って，ScoutView は使いやすいインタフェースを備えたシステムであることが示せた．また，アンケートの意見により,「使いやすく，見やすかった」,「速くできた」,「調べ物に使ってみた」と好印象であった．

しかし，アンケートの結果より,「慣れれば使いやすそう」,「画面が小さい場合は，場所をとられると感じるだろう」という意見も存在した．後者の意見については，現在の傾向としてディスプレイの大型化が進んでおり，画面の大きさは問題にならないと考えている．また，モバイル端末などの画面が特に小さい端末上での実装を行う場合には，それに沿った機能の改変が必要になると考えられる．

継続利用については，表 1 より平均値が 3.4 であり，良い評価となった．

5. ま と め

5.1 結　　論

本研究では閲覧ページ内におけるページ内ナビゲーションスタイルを 2 種類に分類し，閲覧ページが長大な場合，そのスクロールがユーザへの負荷となるという問題を説明した．

その問題点に対して，より閲覧すべき箇所を抽出したインタフェースの提供を目的として，“ScoutView”の設計を行い，Mozilla Firefox の拡張機能として実装，評価を行った．

その結果，本システムは，長大な閲覧ページのページ内ナビゲーションタスクにおいて，時間を短縮することができ，ユーザの負荷を軽減できるインタフェースを備えていることを示すことが出来た．

5.2 今後の課題

5.2.1 検索エンジンとの連携

Web による情報検索タスクにおいて，Google[1] のような検索エンジンを利用して所望の情報の存在するであろうサイトの検索を行う．

その際，検索結果ページ上で，指定した検索クエリをハイライトさせるために，“キャッシュ”のリンクをクリックして，結果ページを開くことがある．このように，検索結果ページから Web ページを開いた際には，その検索クエリで即座に近傍窓を実行できると，所望の情報を素早くに取得できると考えられる．

あるいは，検索 API の利用により，検索結果ページ内の各結果ページのスニペット直下にいくつかの近傍窓を提示することでも，同様の効果が期待できる．

5.2.2 一般的なブラウジングにおける使用

今回の評価では，実験ページにおいて，あくまで一時的な利用に対しての評価を行った．当然，このようなタスクには一般的なブラウジングと異なる部分が存在する．

従って，今後は一般的な Web ページに対してのブラウジングにおいて，システムの利用はユーザへどのような影響を与えていくか，どのようなシナリオの場合に有効であるかについても調査の必要があると考えている．

参 考 文 献

1) Google http://www.google.co.jp/ .
2) Google ツールバー http://toolbar.google.com/T6/intl/ja/ .
3) Mozilla Firefox http://mozilla.jp/ .
4) canvas-MDC https://developer.mozilla.org/ja/HTML/Canvas .
5) Bongwon Suh, Allison Woodruff, Ruth Rosenholtz, and Alyssa Glass. “Popout prism: Adding perceptual principles to overview+detail document interfaces.”. Proc. CHI 2002,pp.251-258.
6) 若松亮太，田中二郎，志築文太郎，三末和男，高橋伸.
“Web ページの概観と更新差分を用いたブラウジング支援”. 筑波大学 第三学群情報学類 卒業研究論文. http://www.iplab.cs.tsukuba.ac.jp/paper/waka_thesis.pdf .
7) P. Baudisch, B. Lee, and L. Hanna. “Fishnet, a fiheye web browser with search term popouts: a comparative evaluation with overview and linear view”. Proc. AVI 2004,pp.133-140.