高解像度ディスプレイを最大限に利用するPDFビューアの開発

* 三浦 元喜
千葉工業大学 工学部 情報通信システム工学科

要旨： 近年販売されているPCモニタの多くは横長パネルである。その一方で，一般的に使われているPDFビューアは，見開き2ページよりも多くのページ枚数を横方向に並べて配置する機能を備えていない。そのためPDFを表示する際，PCモニタの表示能力を十分に活用できなかった。我々は横長パネルを備えたPCモニタを有効活用し，PDFファイルの閲覧タスクの効率化と簡便化によって知的活動の生産性を向上するため，3ページ以上を横方向に配置して閲覧できるPDFビューアを開発した。本稿ではプロトタイプシステムを含む開発の経緯と，試用結果を報告する。

キーワード： 表示装置の活用，電子文書，レイアウト，閲覧インタフェース，知的生産性向上

1 背景と目的

レイアウトを保持した電子文書のフォーマットとして，Portable Document Format (PDF) は広く普及している。また，表示用のソフトウェア（PDFビューア）の多くは無償で利用できるため，論文や技術文書，契約書類，学生のレポートや報告書など，PDF文書を計算機上で閲覧して確認したり，印刷するのに多く用いられている。

既存のPDFビューアの多くは，ページを表示する際のオプションとして「横幅をウィンドウに合わせる」や,「見開き2ページ」を提供している。また，上下方向の連続スクロールの有り無しや，全画面表示，サムネイル表示，しおり表示，検索機能など，計算機上で文書をハイパーテキスト的にジャンプしながら読むための機能を備えている。

ところで，近年のPC用ディスプレイ/モニタは広い表示面積をもったものや，高解像度化によって多くのピクセルを備えた製品が比較的安価で販売されている。フルHD(1920×1080)〜WQHD(2540×1440) はもとより，UWQHD(3440×1440) や 4K(3840×2160) の解像度をもつモニタも普及しつつあり，以前より多くの情報を同時に表示できるようになった。これらのPCモニタの多くは，横長のパネルを搭載している。

しかし，既存のPDFビューアの多くは，本のメタファを踏襲した「見開き2ページ」しか対応していない。そのため，画面の横方向に3ページ以上を配置することができず，近年の横長PCモニタの横長・高解像度パネルを十分に利用できていなかった。また，適切な表示ズーム率の調整に手間がかかるという問題があった。

そこで我々は，PCモニタの横長パネル表示領域を有効活用し，PDF閲覧・確認タスクの効率化と簡便化によって知的生産性の向上を図ることを目的として，3ページ以上を横方向に配置して画面表示できるPDFビューアを開発した。

2 TilePDF

図1に，コンセプトおよびユーザビリティの確認と，必要機能を洗い出すために作成したプロトタイプシステムの画面を示す。プロトタイプシステムは，既存Webシステム (LAMP構成) の拡張として作成した。PDFファイルをサーバにアップロードすると，Poppler-utils[1] パッケージに含まれる pdftoppm コマンドによって，各ページの画像 (PNG) ファイルを生成し，サーバに保存する。閲覧ページを開くと，図1のように各ページの画像が横方向に配置される。また，画面上部に，背景色や横並びページ数，ページごとのマージン量，

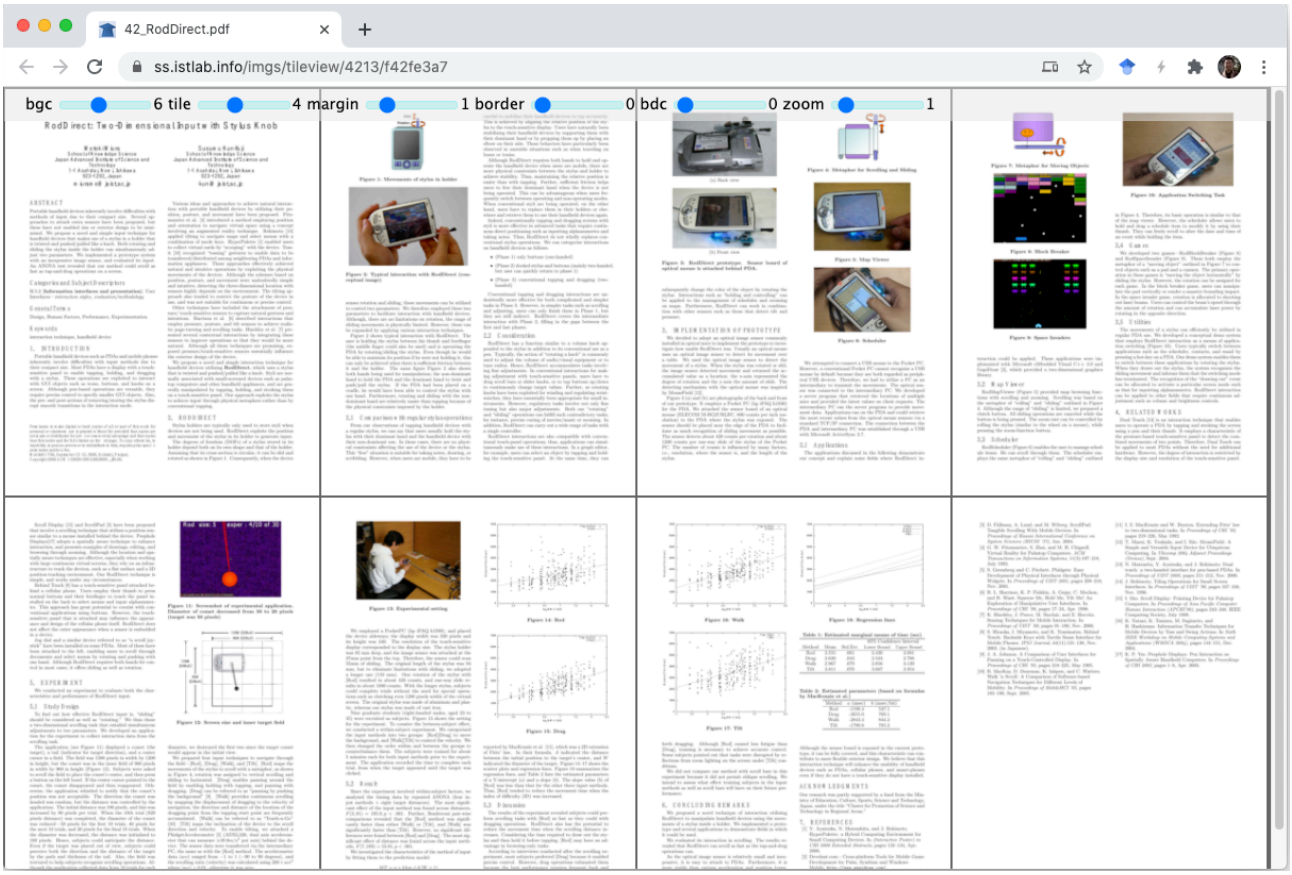

図 1: 初期プロトタイプシステム (サーバで生成した画像を並べて表示する Web ページ)

画像を囲む枠線の幅，枠線の色，ズーム量の調整のためのスライダが表示される。ズーム量の調整を行うと，各ページの表示領域は変化させずに，ページの中央部分を拡大表示するため，ページ周囲の余白部分を非表示にすることができる。これにより，少しでも PDF 文書内容のテキストを読みやすく表示することを意図している。

著者は構築したプロトタイプシステムに論文や文書，PowerPoint スライドを PDF ファイルに変換したものなどを 30 本アップロードして，2 週間ほど日常的な閲覧作業に利用した。その結果，以下の知見が得られた。

- ページ数の多い PDF ファイルを閲覧する際の，スクロールの手間を省くことができた。
- 論文やマニュアル類など，6〜10 ページの PDF 文書について，全体の構成を確認しやすくなった。
- 横に並べるページ数の調整は，頻繁に行うことがわかった。これは，表示を縮小（横並び枚数を増加）して全体の構成を確認したあと，表示を拡大（横並び枚数を減少）して詳細を読むといった要求があったからである。そのためスライダによる操作よりも，ショートカットキーによって直接枚数を指定する操作が適していることがわかった。
- 横に並べるページ数を設定すれば，表示ズーム率を微調整することなく，現在のウィンドウのサイズにぴたっと合うように配置できるため，手間がかからない。Web ブラウザのウィンドウサイズを手動で変更しても，表示可能領域の横幅に合わせて自動的にサイズが調整され，必要に応じてスクロールバーが表示される。既存の PDF ビューアに比べて，表示調整作業が軽減した。

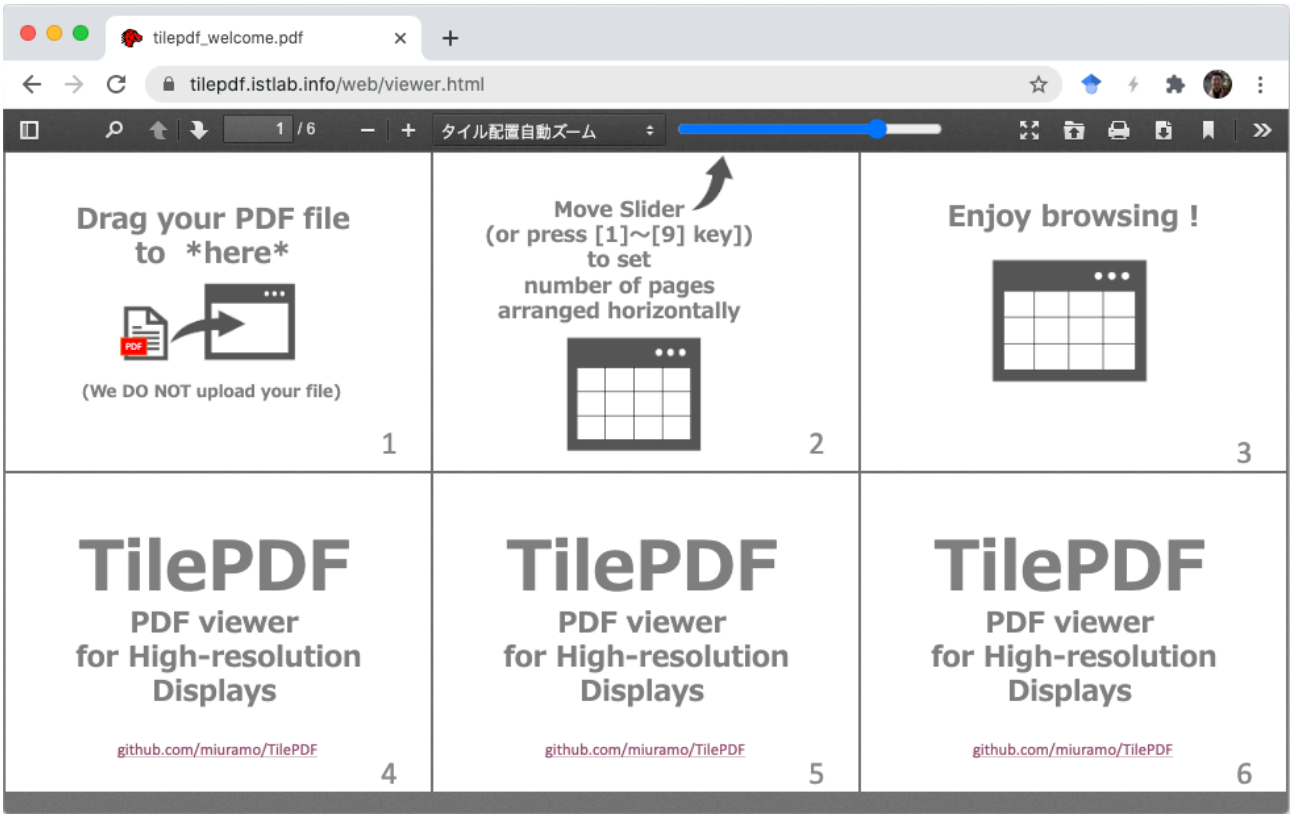

図 2: TilePDF 初期画面

- 背景色やマージン量，画像を囲む枠線の幅，枠線の色の調整機能は，当初は使用されたが，徐々に利用されなくなった。その理由として，白地 PDF 文書を閲覧することが多かったため，細かな表示色パラメータを調整する必要性に乏しかったことが挙げられる。
- ズーム量の調整機能は，PDF 文書によっては余白部分を削減できるため有用だが，あまり利用されなかった。ズーム量を調整すると，画面に表示される余白が少なくなることで，ページの印象が変化し，閲覧時の違和感が増すことが影響していると考えられる。
- PDF ファイルを事前にサーバにアップロードして，画像に変換しておくのは手間がかかるうえ，画像データ容量が保存領域を圧迫する欠点があった。一方で，サーバにアップロードした PDF ページ画像は Web ブラウザでどこからでも参照でき，かつレンダリングが早いという利点があった。

プロトタイプシステムの試用から得られた知見をふまえて，我々は PDF.js[2] を改良・拡張した PDF ビューア TilePDF を開発した。PDF.js は Javascript で記述された PDF ビューアであり，Web ブラウザ上で動作する。TilePDF の初期画面を図 2 に示す。初期画面では，PDF 内容を 3 ページずつ横並びにして表示する。初期画面に表示する PDF は，使用方法を説明する簡易な説明書になっている。利用者は，PDF ファイルを直接ウィンドウ内にドロップするか，右上のボタン (または CTRL+O キー) によってファイル選択画面を表示し，PDF を読み込ませて表示することができる。また 1〜9 の数字キーが，横並びページ数を設定するショートカットキーとなっている。そのため，利用者はスライダを操作しなくても，数字キーで簡単に配置とズーム率を変更できる。なお横並び枚数の設定は，別の PDF ファイルを読み込ませたときも継続される。これは複数の学生レポートを次々に開いて確認していく場面において，操作を

省略できる効果がある。

実装上の工夫について述べる。元の PDF.js の機能を変更せず残すため，我々は表示モードとして「タイル配置自動ズーム」を新たに追加したうえで，横並び枚数を設定するスライダを配置した。なお，スライダは左右反転し，マウスカーソルの動きと，画面拡大/縮小のイメージが一致するようにした。元の PDF.js では画面最大化を行うと，プレゼンテーションモードとなり，1 ページずつ全体表示しクリックで次ページに移動する。しかし，我々は全画面表示時においてもタイル配置表示を維持したかったため，画面最大化を行ったときのモードによって挙動を切り替えるようにした。

3　関連研究と考察

Baudisch らは，Focus plus context screens という高解像小画面の focus area と低解像大画面の context area を同一空間に統合した表示装置を提案し，プロトタイプとして LCD とプロジェクションスクリーンによる構成方法を示している [3]。大画面による周辺視野と，高解像画面による作業領域を歪みなく空間的に接続するため，作業効率が高まることを [4] で検証し確認している。本研究は高解像ディスプレイを活用した PDF ビューアの提案であるが，大画面による周辺視野を活用し，作業効率を高める点では共通している。

本研究は，筆者が大学院生のときの助手の先生の論文読みの慣習から着想を得ている。その先生は論文を読む際，必ず印刷した紙をすべて横長の机に並べたうえで，顔を近づけ，のめり込みながら作業されていた。熱量を持った姿が周囲に与える影響は，三村が [5] で言語化した，KJ 法の作法や態度と重なる部分もある。

4　まとめと今後の課題

我々は 3 ページ以上のページを横並びに配置して表示できる PDF ビューアを提案した。プロトタイプの開発と試用を踏まえ，Javascript ベースの PDF ビューアを改良することにより，当初の目的を達成した。提案手法に対する有効性検証を行うことが今後の課題である。本稿で示した TilePDF は，`https://tilepdf.istlab.info/` から利用できる。またソースコードは [6] で公開している。新型コロナウィルスの影響で電子文書の流通と利用が増えている昨今，ペーパーレス環境での作業の効率化と快適化に少しでも貢献できれば幸いである。

謝辞

本研究の一部は令和 2 年度千葉工業大学公募型重点配分経費の支援によるものです。

参考文献

[1] Poppler project, (年不明) Poppler Website. `https://poppler.freedesktop.org/` (2020 年 8 月 18 日確認)

[2] Mozilla, and individual contributors, (年不明) PDF.js – A general-purpose, web standards-based platform for parsing and rendering PDFs –. `https://mozilla.github.io/pdf.js/` (2020 年 8 月 17 日確認)

[3] Baudisch, P., Good, N. and Stewart, P. (2001) Focus plus Context Screens: Combining Display Technology with Visualization Techniques, Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology, UIST '01, New York, NY, USA, Association for Computing Machinery, pp. 31–40.

[4] Baudisch, P., Good, N., Bellotti, V. and Schraedley, P. (2002) Keeping Things in Context: A Comparative Evaluation of Focus plus Context Screens, Overviews, and Zooming, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '02, New York, NY, USA, Association for Computing Machinery, pp. 259–266.

[5] 三村修 (2005) KJ 法における作法の研究, 北陸先端科学技術大学院大学 修士論文. `http://hdl.handle.net/10119/537`

[6] Miura, M. (2020) TilePDF Github page. `https://github.com/miuramo/TilePDF` (2020 年 8 月 18 日確認)