PDF recompression using JBIG2

Radim Hatlapatka
<208155@mail.muni.cz>

17th March 2010

EuDML
The EUROPEAN DIGITAL MATHEMATICS LIBRARY

Outline

- Motivation
- Standard JBIG2
- PDF recompression
- Jbig2enc and leptonica
- Current achievements
- Planned next steps
- Conclusion

Motivation of this project

- decrease size of PDF documents
- improvement of perceptually lossless compression of encoder jbig2enc by removing flyspecks
- applying to documents containing scanned pages containing mathematical expressions (on DML-CZ and EuDML)

Motivation of this project

- decrease size of PDF documents
- improvement of perceptually lossless compression of encoder jbig2enc by removing flyspecks
- applying to documents containing scanned pages containing mathematical expressions (on DML-CZ and EuDML)

Motivation of this project

- decrease size of PDF documents
- improvement of perceptually lossless compression of encoder jbig2enc by removing flyspecks
- applying to documents containing scanned pages containing mathematical expressions (on DML-CZ and EuDML)

JBIG2

- standard (ISO/IEC 14492) for compression of bi-level images with a very good compression ratio
- designed to compress text in images, i.e. scanned text, faxes
- supports compression of multipage documents by using global dictionary
- supports both lossless and lossy compression (also perceptually lossless compression)
- segments each page to several regions, they are typically three (text region, halftone region and generic region)
- apply specific coding for each region (modifications of arithmetic or Huffman coding)
- supported in PDF since version 1.4

JBIG2

- standard (ISO/IEC 14492) for compression of bi-level images with a very good compression ratio
- designed to compress text in images, i.e. scanned text, faxes
- supports compression of multipage documents by using global dictionary
- supports both lossless and lossy compression (also perceptually lossless compression)
- segments each page to several regions, they are typically three (text region, halftone region and generic region)
- apply specific coding for each region (modifications of arithmetic or Huffman coding)
- supported in PDF since version 1.4

JBIG2

- standard (ISO/IEC 14492) for compression of bi-level images with a very good compression ratio
- designed to compress text in images, i.e. scanned text, faxes
- supports compression of multipage documents by using global dictionary
- supports both lossless and lossy compression (also perceptually lossless compression)
- segments each page to several regions, they are typically three (text region, halftone region and generic region)
- apply specific coding for each region (modifications of arithmetic or Huffman coding)
- supported in PDF since version 1.4

JBIG2

- standard (ISO/IEC 14492) for compression of bi-level images with a very good compression ratio
- designed to compress text in images, i.e. scanned text, faxes
- supports compression of multipage documents by using global dictionary
- supports both lossless and lossy compression (also perceptually lossless compression)
- segments each page to several regions, they are typically three (text region, halftone region and generic region)
- apply specific coding for each region (modifications of arithmetic or Huffman coding)
- supported in PDF since version 1.4

JBIG2

- standard (ISO/IEC 14492) for compression of bi-level images with a very good compression ratio
- designed to compress text in images, i.e. scanned text, faxes
- supports compression of multipage documents by using global dictionary
- supports both lossless and lossy compression (also perceptually lossless compression)
- segments each page to several regions, they are typically three (text region, halftone region and generic region)
- apply specific coding for each region (modifications of arithmetic or Huffman coding)
- supported in PDF since version 1.4

JBIG2

- standard (ISO/IEC 14492) for compression of bi-level images with a very good compression ratio
- designed to compress text in images, i.e. scanned text, faxes
- supports compression of multipage documents by using global dictionary
- supports both lossless and lossy compression (also perceptually lossless compression)
- segments each page to several regions, they are typically three (text region, halftone region and generic region)
- apply specific coding for each region (modifications of arithmetic or Huffman coding)
- supported in PDF since version 1.4

JBIG2

- standard (ISO/IEC 14492) for compression of bi-level images with a very good compression ratio
- designed to compress text in images, i.e. scanned text, faxes
- supports compression of multipage documents by using global dictionary
- supports both lossless and lossy compression (also perceptually lossless compression)
- segments each page to several regions, they are typically three (text region, halftone region and generic region)
- apply specific coding for each region (modifications of arithmetic or Huffman coding)
- supported in PDF since version 1.4

What is halftone

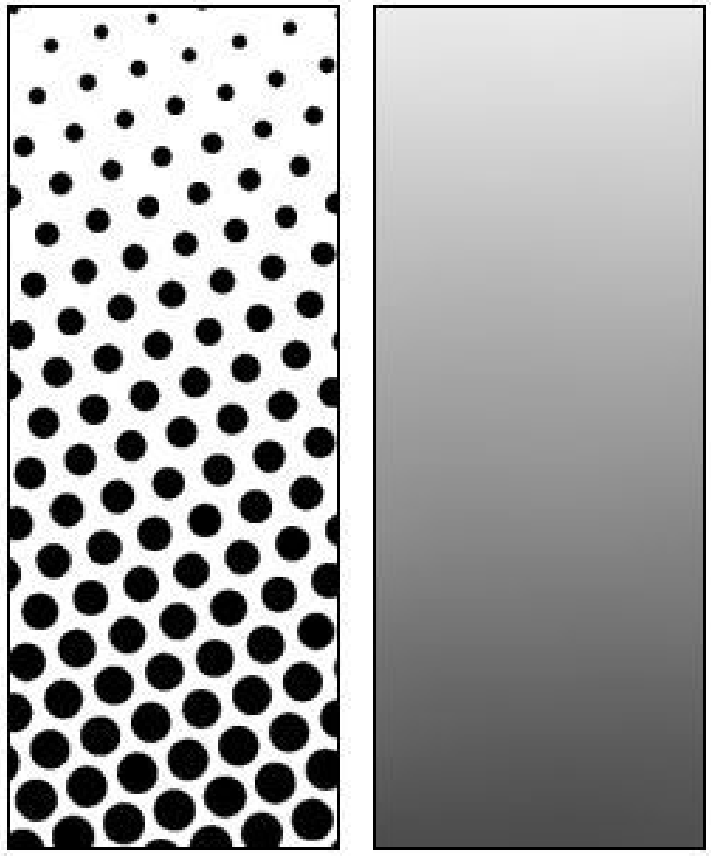


Figure: Picture of how halftone works[1]

[1] *Extracted from <http://encyclopedia.tfd.com/halftone> 8. 2. 2010*

Example of how is saved JBIG2 image in PDF

```
 2 0 obj
«
/DecodeParms
« /JBIG2Globals 1 0 R »
/Width 3265
/BitsPerComponent 1
/Height 4911
/Filter /JBIG2Decode
/Subtype /Image
/Length 4582
/ColorSpace /DeviceGray
/Type /XObject
»
stream
....
endstream
```

PDF recompressor

- idea in steps:
  1. extraction of image bit streams and conversion to format accepted by encoder jbig2enc
  2. invoking of encoder on extracted images
  3. replacing images by their compressed version according to JBIG2 standard
     - input of concrete image
     - input of global data and making reference to it (global data put only once)
- used libraries
  - PDFBox – used for extraction of images (extraction including conversion)
  - IText – used for decryption of PDF document and replacing original images their compressed version

PDF recompressor

- idea in steps:
  1. extraction of image bit streams and conversion to format accepted by encoder jbig2enc
  2. invoking of encoder on extracted images
  3. replacing images by their compressed version according to JBIG2 standard
     - input of concrete image
     - input of global data and making reference to it (global data put only once)
- used libraries
  - PDFBox – used for extraction of images (extraction including conversion)
  - IText – used for decryption of PDF document and replacing original images their compressed version

PDF recompressor

- idea in steps:
  1. extraction of image bit streams and conversion to format accepted by encoder jbig2enc
  2. invoking of encoder on extracted images
  3. replacing images by their compressed version according to JBIG2 standard
     - input of concrete image
     - input of global data and making reference to it (global data put only once)
- used libraries
  - PDFBox – used for extraction of images (extraction including conversion)
  - IText – used for decryption of PDF document and replacing original images their compressed version

PDF recompressor

- idea in steps:
  1. extraction of image bit streams and conversion to format accepted by encoder jbig2enc
  2. invoking of encoder on extracted images
  3. replacing images by their compressed version according to JBIG2 standard
     - input of concrete image
     - input of global data and making reference to it (global data put only once)
- used libraries
  - PDFBox – used for extraction of images (extraction including conversion)
  - IText – used for decryption of PDF document and replacing original images their compressed version

PDF recompressor

- idea in steps:
  1. extraction of image bit streams and conversion to format accepted by encoder jbig2enc
  2. invoking of encoder on extracted images
  3. replacing images by their compressed version according to JBIG2 standard
     - input of concrete image
     - input of global data and making reference to it (global data put only once)
- used libraries
  - PDFBox – used for extraction of images (extraction including conversion)
  - IText – used for decryption of PDF document and replacing original images their compressed version

PDF recompressor

- idea in steps:
  1. extraction of image bit streams and conversion to format accepted by encoder jbig2enc
  2. invoking of encoder on extracted images
  3. replacing images by their compressed version according to JBIG2 standard
     - input of concrete image
     - input of global data and making reference to it (global data put only once)
- used libraries
  - PDFBox – used for extraction of images (extraction including conversion)
  - IText – used for decryption of PDF document and replacing original images their compressed version

Jbig2enc and leptonica

- open-source JBIG2 encoder [4]
- uses open-source library leptonica [1] for manipulation with images and bitmaps of symbols
- supports only arithmetic coding

Jbig2enc and leptonica

- open-source JBIG2 encoder [4]
- uses open-source library leptonica [1] for manipulation with images and bitmaps of symbols
- supports only arithmetic coding

Jbig2enc and leptonica

- open-source JBIG2 encoder [4]
- uses open-source library leptonica [1] for manipulation with images and bitmaps of symbols
- supports only arithmetic coding

Modification of jbig2enc

- comparision of all templates (representative symbols) with the same size for finding equivalence
  - two templates are considered equivalent if there is not found big enough accumulation of differences
  - we look for accumulations in shapes such as points or lines
- unification of two equivalent symbols to one

Modification of jbig2enc

- comparision of all templates (representative symbols) with the same size for finding equivalence
  - two templates are considered equivalent if there is not found big enough accumulation of differences
  - we look for accumulations in shapes such as points or lines
- unification of two equivalent symbols to one

Preliminary results

These are average results from aplying to 810 documents

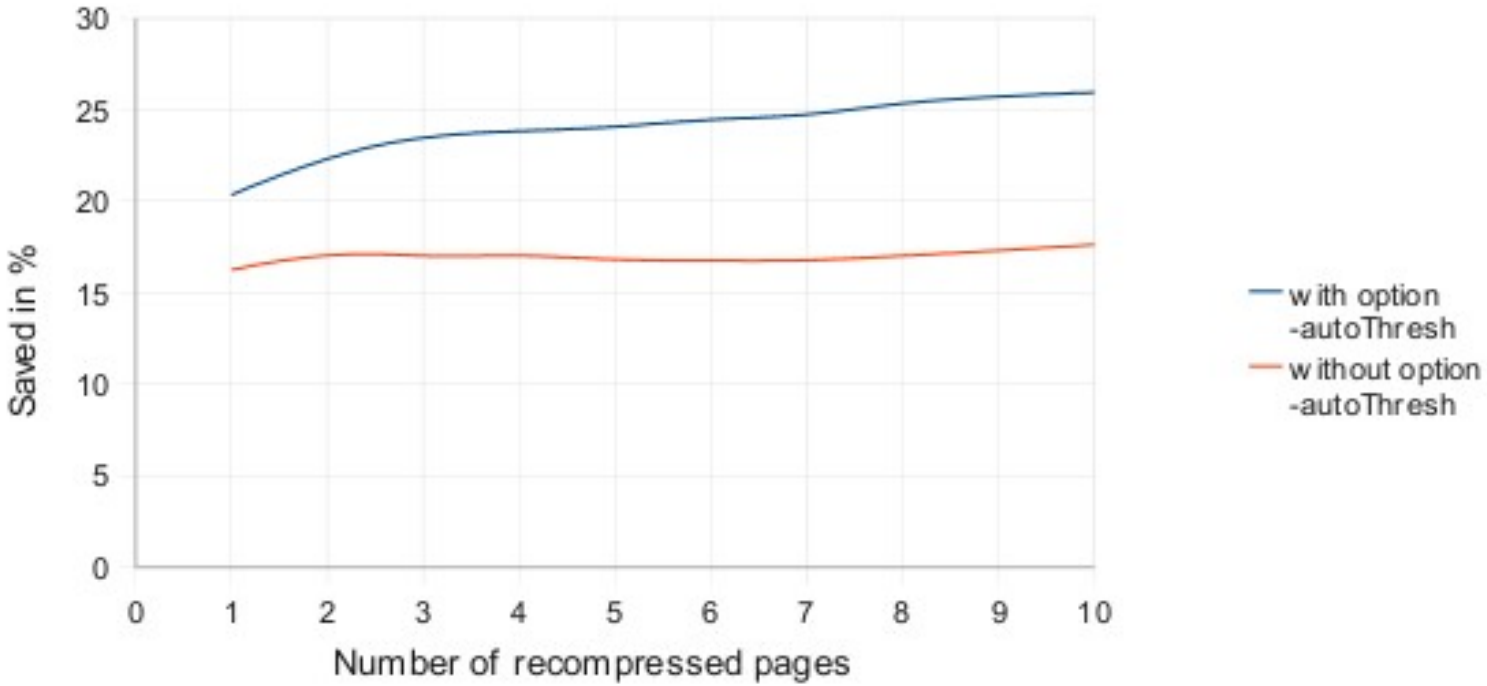


Figure: Comparision of reductions in size through different number of pages

Problematic comparision

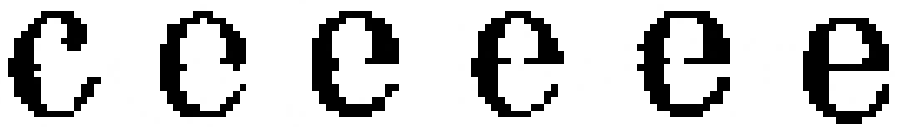


Figure: Example of problematic comparision

Current and future steps

- OCR tools and techniques
  - interface for calling InftyReader[1] that will return up to ten results containing recognised letter with score (accuracy of a hit)
  - applying some other procedures on results that we got like featured vectors to decide which to choose
- heuristics to decrease number of compared symbols
  - suggestions are welcome
- PDF recompressor
  - improve posibilities such as recompression of images already compressed according to JBIG2 standard
- combination with pdfsizeopt.py [6] developed by Péter Szábo (Google)

[1] info about InftyProject can be found at <http://www.inftyproject.org/>

Current and future steps

- OCR tools and techniques
  - interface for calling InftyReader[1] that will return up to ten results containing recognised letter with score (accuracy of a hit)
  - applying some other procedures on results that we got like featured vectors to decide which to choose
- heuristics to decrease number of compared symbols
  - suggestions are welcome
- PDF recompressor
  - improve posibilities such as recompression of images already compressed according to JBIG2 standard
- combination with pdfsizeopt.py [6] developed by Péter Szábo (Google)

[1] info about InftyProject can be found at <http://www.inftyproject.org/>

Current and future steps

- OCR tools and techniques
  - interface for calling InftyReader[1] that will return up to ten results containing recognised letter with score (accuracy of a hit)
  - applying some other procedures on results that we got like featured vectors to decide which to choose
- heuristics to decrease number of compared symbols
  - suggestions are welcome
- PDF recompressor
  - improve posibilities such as recompression of images already compressed according to JBIG2 standard
- combination with pdfsizeopt.py [6] developed by Péter Szábo (Google)

[1] info about InftyProject can be found at <http://www.inftyproject.org/>

Current and future steps

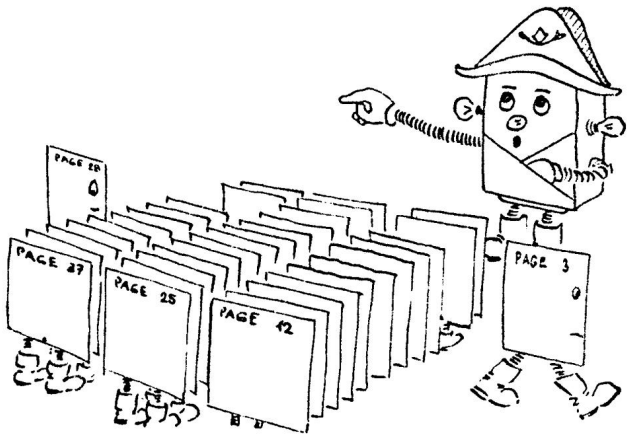
- OCR tools and techniques
  - interface for calling InftyReader[1] that will return up to ten results containing recognised letter with score (accuracy of a hit)
  - applying some other procedures on results that we got like featured vectors to decide which to choose
- heuristics to decrease number of compared symbols
  - suggestions are welcome
- PDF recompressor
  - improve posibilities such as recompression of images already compressed according to JBIG2 standard
- combination with pdfsizeopt.py [6] developed by Péter Szábo (Google)

[1] info about InftyProject can be found at <http://www.inftyproject.org/>

Conclusion

- prototype is already functional and tested at sample taken from data on DML-CZ
- still much to do to achieve optimal compression ratio of perceptually lossless compression
- suggestions are welcome

Questions?



References

Dan Bloomberg:
Leptonica.
<http://www.leptonica.com/>.

R. Hatlapatka:
JBIG2 compression.
Brno 2010. Bachaleor thesis at Faculty of Informatics MU. Advisor doc. RNDr. Petr Sojka, Ph.D.
http://is.muni.cz/th/208155/fi_b/.

R. Hatlapatka:
Websites of the project PDF recompression.
http://www.fi.muni.cz/~xhatlap/pdfRecompression.html.

Adam Langley:
Jbig2enc.
<http://github.com/agl/jbig2enc/>.

Masakazu Suzuki:
Infty project.
Faculty of Mathematics and Graduate School of Mathematics Kyushu University, 2008.
<http://www.inftyproject.org/>.

Péter Szábo:
Optimizing PDF output size of TEX documents.
<http://code.google.com/p/pdfsizeopt/>.