Document Image Analysis with Leptonica

*Phototech EDU, 4 April 07*

Dan Bloomberg
bloomberg@ieee.org
Introduction

- Image analysis in a course in photographic technology?
- Image analysis in the last century.
- Hofstadter’s 100 milliseconds and image processing.
- Trade-off between speed and accuracy.
- Two examples of scaling
  - Linear interpolation on color
  - Rank order cascade of 2x reductions on binary
- Why document image analysis?
  - Easier than natural scenes
  - Useful: conversion from paper to digital
  - Interesting: input is not well-defined
Roadmap
Outline of talk

- Goals
  - Page information extraction
  - Restoration and/or appearance improvement
  - Compression

- Approach
  - Nonlinear/Shape and Texture/Use the image

- Primary tools
  - Image morphology
  - Affine transforms
  - Counting and components
  - Seedfill
  - Leptonica library

- Example applications
  - Page image segmentation
  - Background cleaning of bad photocopy
  - Skew, keystoning and baselines
  - Unsupervised shape classification
  - Color segmentation/quantization
Goals
Page information extraction

- Global information
  - Skew and text orientation
  - Non-affine warping (e.g., projective)

- Components on the page
  - Text, image, rules, ...
  - What are they?
  - Where are they located?
  - What is the hierarchical arrangement?
  - What are the equivalence classes?

- Photometry
  - What is the background color?
  - Are there color images?
Restoration and/or appearance improvement

- Geometrical
  - Image deskew
  - Global dewarping

- Color mapping
  - Set background to uniform color
  - Compensate for lighting variations
  - Map text to increase contrast; preserve antialiasing
  - Map images for larger dynamic range
  - Detect and remove color moire

- Other
  - Remove noise from binary scans
  - Remove bleedthrough
  - Scale to gray for display
  - Interpolated upscaling for print
  - Quantization for compression
Compression

- **Artifacts**
  - JPEG 8x8 block noise near text
  - Color moire: alias on halftones and gravure
  - Binary thresholding
    - Increases contrast: bad for images
    - Removes antialias: bad for text at low resolution

- **Avoidance techniques**
  - Uniform background
  - Quantization of text
  - Capture at higher resolution
  - Demosaic to gray if no color
  - Mixed raster output
Approach
Approach

- Nonlinear: decisions made on each pixel
  - Linear operations don’t make decisions
  - Implicit labels assigned to pixels
  - Bottom-up aggregation

- Extraction of shape and texture
  - Shape at one scale is texture at another
  - Work at appropriate scale
  - Use morphology to sieve
  - Use morphology and rank reductions to modify texture
  - Use seedfill for robust segmentation and labelling

- Image as primary representation
  - All the information is there – don’t lose it
  - Use image processing to do (nearly) everything
  - Complex, difficult and limiting to use other representations
  - Simple, easy and general to visualize imaging methods
Primary tools
Image morphology (1)

- References
  
  www.leptonica.org/binary-morphology.html
  www.leptonica.org/papers/morphdefs.pdf

- What is it?
  
  Method for extracting shape and texture
  Image processing operations: dilation and erosion
  Analogy with convolution
  Nonlinear: special case of rank order filters
    - Dilation is MAX, Erosion is MIN
  Kernel is Sel ("structuring element")
    - Hits, misses, don’t-cares, origin
  Opening and closing are composite operations
    - idempotent; independent of origin
  Dualities
  Hit-miss operation is general pattern match
Image morphology (2)

- Historical
  Invented in France in the 60s
  Very slow adoption in the US
- Example of hit-miss Sels

These are used to identify character ascenders and descenders
Image morphology (3)

- Implementation through rasterop
  - Always use packed images and full word operations
  - Conceptual: test Sel at each point on src
  - Actual: let Sel direct full image rasterops
    - Erosion: copy first; then AND
    - Dilation: OR each hit
  - Efficiency for brick Sels
    - Separable in x and y
    - Composable as sequence at different scales

- Implementation through dwa (dest word accumulation)
  - Auto-gen’d code
  - Unrolled destination word loop
    - Typically 3-4x faster than rasterop

- Both can be invoked for brick Sels with an interpreter.
Affine transforms (1)

- Translation: rasterop
- Shear: rasterop
- Rotation
  
  Reference: [www.leptonica.org/rotation.html](http://www.leptonica.org/rotation.html)
  
  By rasterop: 2 shear and 3 shear
  
  By area mapping (linear interpolation)

- Scaling
  
  Reference: [www.leptonica.org/scaling.html](http://www.leptonica.org/scaling.html)
  
  Useful for many things
  
  Rendering: interpolation up; antialias down
  
  Combining with depth change for rendering
  
  Choosing scale at which to work
  
  Combining morphology with subsampling: texture filtering
Affine transforms (2)

- Scaling types
  - Binary to gray (downscale)
    - example: display high res binary on screen as grayscale
  - Gray to binary (upscale)
    - example: convert to high res binary for print, display
  - Gray to gray
  - Binary to binary

- Binary to binary: rank order 2x cascade
  - Generalization of morphology + subsampling
  - Useful for texture filtering
  - Fast word parallel operation
  - Rank = 1 (1 or more are fg) solidifies fg
  - Rank = 4 (all 4 are fg) erodes fg
Counting and components

- **Fg pixels in 1 bpp images**
  - Test for *any* fg pixels
  - Sum pixels on raster scanlines
  - Use for determining skew

- **Connected components in 1 bpp images**
  - Use for labeling components
  - Use for adaptive thresholding; e.g., word segmentation

- **Histograms in 8 bpp images**
  - Attach tentative labels (text, image)
  - Generate 1 bpp masks
Seedfill

- Use to label connected components
  - Remove components sequentially
  - Optionally save component bitmap

- Requires seed and mask images
  - Fill into seed; clip to mask

- Slow, parallel, morphological method
  - Iterate with 3x3 brick Sel for 8-c.c. fill
  - Number of iterations depends on component size

- Fast, sequential, raster/anti-raster fill
  - Use for all full-image seedfill
  - Typically requires several pairs of traverses
  - Number of iterations is independent of component size

- Grayscale version exists
  - Fast, sequential, raster/anti-raster fill
  - Use for analyzing peaks
Leptonica library (1)

- Lightweight (efficient) C library
  Mostly low-level imaging functions
  Written in 2001 - 2003; maintained to present
  Works with both endians
  About 20 structs, 1000 functions
  Open source
  Most parts have been extensively tested
  Tailored for document image analysis
  The image is the primary object

Available at:
- www.leptonica.org
- code.google.com/p/leptonica
- debian packages: libleptonica, etc.
Leptonica library (2)

- Basic infrastructure
  - rasterop (depth independent)
  - affine transforms
    - scaling, translation, rotation, shear
    - on all depths; often with or without colormaps
  - binary morphology (two different implementations)
  - grayscale morphology and convolution
  - connected components and sequential seedfill
  - transforms combining changes in scale and pixel depth
  - pixelwise masking, blending, enhancement, arith ops, etc.
  - I/O for jpeg, png, tiff, pnm, bmp; O for PostScript
  - lots more
- Various "applications"
  octcube-based color quantization (incl. dithering)
  skew determination of doc images
  segmentation of page images with mixed text/images
  jbig2 unsupervised classifier
  border representations of bitmaps; raster conversion
  PostScript wrapping of images (levels 1,2)
  playing around (e.g., least-cost paths in images)
Example Applications
Page segmentation (1)

- First identify halftone image regions
- Then identify text lines
- Then aggregate into text blocks
Page segmentation (2)
Page segmentation (3)
Page segmentation (4)

- pixt1 = pixReduceRankBinaryCascade (pixs, 4, 4, 3, 0);
- pixt2 = pixOpenBrick (NULL, pixt1, 5, 5);
- pixhs = pixExpandBinary (pixt2, 8);
pixm = pixCloseSafeBrick(NULL, pixs, 4, 4);
Page segmentation (6)

- `pixhm = pixSeedfillBinary(NULL, pixhs, pixm, 4);
  # open to remove small lines, etc.
- `pixOpenBrick (pixhm, pixhm, 10, 10);`
\texttt{pixtext = pixSubtract (NULL, pixs, pixhm);}
- `pixinv = pixInvert (NULL, pixs);`
- `pixvws = pixMorphCompSequence (pixinv, "o5.1 + 01.200", 0);`
Page segmentation (9)

- pixt1 = pixMorphCompSequence(pixinv, "o80.60", 0);
- pixSubtract (pixvws, pixvws, pixt1);
- pixDestroy (&pixt1);
Page segmentation (10)

- pixt1 = pixCloseSafeBrick(NULL, pixs, 30, 1);
Page segmentation (11)

- `pixlines = pixSubtract (NULL, pixt1, pixvws);`
- `pixOpenBrick (pixlines, pixlines, 3, 3);`
Boxa *boxa = pixConnComp
(pixlines, &pixa, 8);

pixGetDimensions
(pixlines, &w, &h, NULL);

pixc = pixaDisplayRandomCmap(pixa,
w, h);

pixcmapResetColor
(pixGetColormap(pixc), 0,
255, 255, 255);
Background cleaning of bad photocopy (1)

- Adaptive background normalization
  - More flexible than background thresholding
  - Two methods to get background values
    - Morphological closing to remove foreground
    - Tiling, bg estimation, filling, smoothing
  - Map pixel values locally
    - Background goes to fixed global value

- Threshold to get binary output if desired

- Simple method for computing background

  pixs = pixRead ("contrast-orig-60.jpg");
  pixt1 = pixCloseGray (pixs, 11, 11);
  or: pixt1 = pixScaleGrayMinMax (pixs, 11, 11, L_CHOSE_MAX);
  pixt2 = pixBlockconv (pixt1, 15, 15);
The Zero-Frequency Problem: Estimating the Probabilities of Novel Events in Adaptive Text Compression

D. W. West and Timothy C. Bell

Abstract - The zero-frequency problem is the problem of estimating the probabilities of previously unseen events. The performance of an adaptive text compression technique is limited by the probability assigned to previously unseen events, or zero-frequency events. We propose a method of $P_{min}$ estimation for use with adaptive text compression. The method is based on the assumption that previously unseen events are more likely to occur in the text than would be expected from a probability distribution. We demonstrate a method for estimating the probability of novel events, and present results showing that our method improves the compression performance of adaptive text compression algorithms.

1. Introduction

The data compression techniques that are currently used to compress text data suffer from a problem known as the zero-frequency problem. This problem arises because the probability distribution used to compress the text data is based on the frequency of occurrence of the text data. Since the probability distribution is based on the frequency of occurrence, it is not able to accurately predict the probability of new events that may occur in the text data. This problem is known as the zero-frequency problem because the probability distribution is not able to accurately predict the probability of events that occur infrequently.

To address this problem, we propose a method for estimating the probability of novel events. Our method is based on the assumption that previously unseen events are more likely to occur in the text than would be expected from a probability distribution. We demonstrate a method for estimating the probability of novel events, and present results showing that our method improves the compression performance of adaptive text compression algorithms.

2. Related Work

There has been a great deal of research on the zero-frequency problem. Many techniques have been proposed to address this problem, including the use of a mixture of probability distributions, the use of a fixed probability distribution, and the use of a model that incorporates the zero-frequency problem. However, none of these techniques have been able to fully address the zero-frequency problem.

Our method is based on the assumption that previously unseen events are more likely to occur in the text than would be expected from a probability distribution. We demonstrate a method for estimating the probability of novel events, and present results showing that our method improves the compression performance of adaptive text compression algorithms.

3. Experimental Results

We have conducted a series of experiments to evaluate the performance of our method. Our results show that our method improves the compression performance of adaptive text compression algorithms.

4. Conclusion

In conclusion, we have proposed a method for estimating the probability of novel events. Our method is based on the assumption that previously unseen events are more likely to occur in the text than would be expected from a probability distribution. We have demonstrated the effectiveness of our method through a series of experiments.

References


The Zero-Frequency Problem: Estimating the Probabilities of Novel Events in Adaptive Text Compression

Lin H. Wataru and Timothy C. Bell

Abstract — The zero-frequency problem is the problem of estimating the probability of novel events in adaptive text compression. This problem is important in adaptive compression algorithms, which are designed to work efficiently for a wide range of data sources. The zero-frequency problem arises when a particular event has never been seen before in the data source. In such cases, the adaptive compression algorithm must estimate the probability of the event based on the data it has seen so far. This estimation is crucial for the performance of the compression algorithm, as it affects the efficiency of the compression algorithm.

Introduction

Adaptive compression algorithms are widely used in various applications, such as data compression, communication, and storage. These algorithms are designed to work efficiently for a wide range of data sources, including text, images, and audio. However, one of the main challenges in adaptive compression is the zero-frequency problem, which occurs when a particular event has never been seen before in the data source. In such cases, the adaptive compression algorithm must estimate the probability of the event based on the data it has seen so far. This estimation is crucial for the performance of the compression algorithm, as it affects the efficiency of the compression algorithm.

Existing solutions to the zero-frequency problem include fixed probability models, which assume that the probability of an event is the same across all data sources, and mixture models, which combine several probability models to estimate the probability of an event. However, these solutions are not always effective, as they may not accurately estimate the probability of rare events.

In this paper, we propose a new method for estimating the probability of novel events in adaptive text compression. Our method is based on a combination of fixed probability models and mixture models, which allows for a more accurate estimation of the probability of rare events. We evaluate the performance of our method using a variety of data sources, and our results show that our method outperforms existing solutions.

Conclusion

In summary, the zero-frequency problem is a significant challenge in adaptive text compression. We have proposed a new method for estimating the probability of novel events in adaptive text compression, which combines fixed probability models and mixture models. Our method is more effective than existing solutions, and we believe that it will be a valuable tool for adaptive text compression.

Acknowledgments

This work was supported in part by the National Science Foundation under Grant No. 0621503.

References


Deskew by differential line sums (1)

- References
  - www.leptonica.org/skew-measurement.html (general background)
  - www.leptonica.org/papers/docskew.pdf (technical description)
- Most robust method (Postl, 1988)
- Use vertical shear to mimic rotation
- Maximize variance of difference of line sums on adjacent lines
- Use coarse linear search followed by binary search
- Typically compute at 100 - 150 ppi resolution
- Accuracy approximately 1 vertical pixel: $1/w$ in radians
- This is about 0.05 degree
- People do not notice angles less than about 0.2 degree
Deskew by differential line sums (2)

Binary search. Variance of diff of ON pixels vs. angle
Deskew by differential line sums (3)
Keystoning and baselines (1)

- `Pix *pix = pixDeskewLocal("keystone.png", 10, 0, 0, 0.0, 0.0, 0.0);`
  - Find local skew in horizontal slices
  - Fit the skew(y) to a straight line
  - Compute the 8-pt projective transform
  - Deskew using the transform

- `Numa *na = pixFindBaselines(pix, &pta);`
  - The Numa gives the baseline (y) for each textline
  - The Pta gives left and right ends of each textline
  - These are used to display the baselines
"I wouldn’t ask that of you," said Ernst, with a laugh. "Even though it is Prince Suvaroff’s country, too?"

"There are Germans you do not like, I suppose—who are even your enemies," said Fred. "Yet now you will forget all that, will you not?"

"God helping us, yes!" said Ernst. "You are right. Your heart must be with your own. But you don’t seem like a Russian, or I would not be helping you."

Then Fred was off, going on his way into the darkness alone. Ernst had told him which road to follow, telling him that if he stuck to it he would not be likely to run into any troop movements.

"Don’t see too much. That is a good rule for one who is in a country at war," he had advised. "If you know nothing, you cannot tell the enemy anything useful, and there will be less reason for our people to make trouble for you. Your only real danger lies in being taken for a spy. And if you are careful not to learn things, that will not be a very great one."
Keystoning and baselines (3)

![Graph showing skew as a function of y.](image)

- **y** (in raster lines from top) range from 0 to 1800.
- **Angle (in degrees)** range from -0.4 to -2.2.
- The graph includes a linear LSF fit and actual data points.

---

Document Image Analysis with Leptonica

Dan Bloomberg, April 4, 2007 - p. 43/52
Keystoning and baselines (4)
Unsupervised shape classification

- General reference: www.leptonica.org/jbig2.html
- Identifies connected components (e.g., characters) in 1 bpp images
- Places them in equivalence classes
- Can also make classes of words (e.g., *dimsum*)
- Can use either correlation or rank hausdorff for decision
- Aggregates components over multiple pages
- This is used in Adam Langley’s JBIG2 open source encoder
  www.imperialviolet.org/jbig2.html
- Must be careful with baselines
- The JBIG2 encoder was used to generate PDFs for Google Book Search
  www.leptonica.org/papers/google-books-pdf.pdf
Color quantization and color segmentation (1)

- Why color quantization?
  Need few levels for text
  Better compression
  Impressionist artwork
  Can use for color seg.

- Octcube is efficient method
  Populate at different depths
  Fast lookup for quantization

- Dither for rendering accuracy (not MSE)

- Generating a colormap vs. quantizing to a colormap
Color quantization and color segmentation (2)

- Fixed levels; depth 2
  - 27 colors

- Fixed levels; depth 3
  - 86 colors
Color quantization and color segmentation (3)

- 256 cells (3,3,2); no dithering
- 56 colors

- 256 cells (3,3,2); dithered
- 81 colors
Color quantization and color segmentation (4)

- 2-pass octree; no dithering
- 174 colors

- 2-pass octree; dithered
- 190 colors
Color quantization and color segmentation (5)

- color segmentation
- 2 colors

- color segmentation
- 3 colors
Color quantization and color segmentation (6)

- color segmentation
- 5 colors

- color segmentation
- 6 colors
Leptonica library extras

- Programmatic interface to gnuplot
- Simple bitmap font facility
- Blending images and simple line graphics
- Generating outlines from rasters and raster conversion from outlines
- Number and string arrays, heaps, stacks, queues, lists, etc.
- Octree color quantization
- Parser to extract C prototypes for a header file
- A large number of regression tests and example programs.