Document Image Decoding

Kris Popat
Xerox PARC

In collaboration with Henry Baird, Dan Bloomberg, Tom Breuel, Dan Greene, Tom Minka (MIT), Justin Romberg (Rice Univ.), Prateek Sarkar, and Taku Tokuyasu (U.C. Berkeley)

November 1, 2000

`these viewgraphs:  ftp://ftp.parc.xerox.com/pub/popat/DID20001101.ps`

Document Image Decoding

- Main reference: Kopec and Chou, IEEE T-PAMI, June 1994 (handout)
- Markov-model based text recognition in document images
- High accuracy; potential for automatic extraction of tags
- Not a robust, hardened system; still requires an expert

Today's Talk

- DID basics
- Speed enhancements
- Extension to grayscale images
- Adding linguistic constraints
- Research challenges

An Extreme Example

- Degraded, subsampled grayscale textline image:

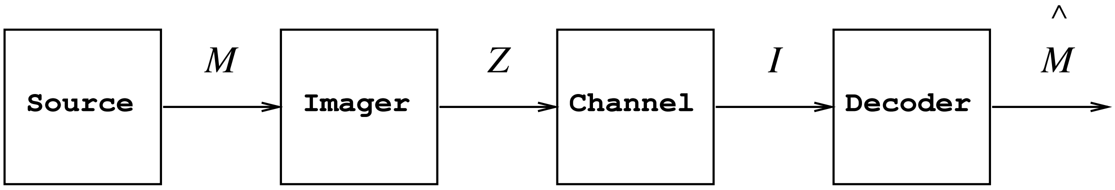
- Result of Document Image Decoding:

`WHITE KITTEN HAD BEEN HAVING ITS FACE WASNED BY THE OLD CAT FOR`

- No prior segmentation required
- Based on the most favorable conditions possible:
  - Accurate degradation model
  - Font metrics and bitmaps known perfectly
  - Restricted alphabet and spacing model
  - Characters guaranteed not to overlap

DID Recognition Framework: Abstract Setting

- Communication systems view:



- Posterior probability of message, given observed image:

$$\begin{aligned} P(M|I) &\propto \sum_Z P(M)P(Z|M)P(I|Z) \\ &= P(M)P(I|M) \end{aligned}$$

- Seek MAP decoding (why?):

$$\hat{M} = \arg\max_M P(M|I)$$

DID Framework Elements

- Imager: A Markov Source
  - There is always a start and stop state
  - Interior states correspond to distinct modes of text generation
  - Printing happens on transitions
  - Each transition t labeled by $(P(t), \Delta_t, Z_t, M_t)$
  - "Running" the source produces a complete path $\pi \longrightarrow (M_\pi, Z_\pi)$
  - $P(\pi)$ induces $P(M, Z)$

DID Framework Elements (Cont.)

- Different paths can result in the same message:

$$P(M) = \sum_{\pi : M_\pi = M} P(\pi)$$

$$P(M, Z) = \sum_{\pi : M_\pi = M} P(Z|\pi)P(\pi)$$

- Viterbi approximation in MAP decoding
- For certain sources, distinct paths always result in distinct messages and images, so in such cases the Viterbi approximation is exact

DID Framework Elements (Cont.)

- Match of hypothesis M' to image is measured by likelihood $P(I|M')$
  - Need to be able to factor into independent terms involving individual transitions
  - Individual terms should be easy to compute
  - Individual terms label edges in trellis

DID Framework Elements (Cont.)

- Trellis (one-dimensional; assumes text-line decoding)
  - Unrolling of the path along the horizontal dimension
  - There is a node for every horizontal pixel position
  - There is an edge for every possible transition (character placement)
- Match scores
  - Edges are labeled by a score that measures the degree of match of the corresponding character in that location
  - Match scores reflect the contribution of the edge to overall likelihood
  - Edge scores must not depend on which other edges appear on the path
  - Motivates disjoint support assumption: foreground pixels can always be assumed to displace background pixels

DID Framework Elements (Cont.)

- Channel model $P(I|Z)$:
  - Independent asymmetric bit-flip model
  - Multilevel templates: bit-flip parameters depend on position in template (Kopec, 1997)
  - Multilevel templates attempt to account for sampling variability and unequal discrimination value of pixels
  - Generalization: Templates are patterns of arbitrary probability distributions (Niles, 1998)

DID Framework Elements (Cont.)

- Aspects Specific to Two-Dimensional DID
  - Markov source structure is critical and layout-dependent
  - Multiple paths for the same message is not unusual (affects Viterbi approximation)
  - Cycles are possible; scheduling difficult
  - Separable Markov source structure offers computational advantages (Kam 1995, Kam and Kopec 1996)

Speed Enhancements

- Vertical Iterated Complete Path Search (Kam and Kopec, 1996)
  - Applicable in the presence of separable two-dimensional source models
  - Shifts computational burden to within-line decoding
- Within-line Iterated Complete Path Search (Minka 1998, Minka et al., 2000 (handout))
- Incremental Viterbi: reuse of path portions across iterations (Ibid.)

Speed Enhancements (Cont.)

- Within-line ICP algorithm:

  Step 1: Constuct trellis, labelling each edge with a cheap upper bound on the match score
  Step 2: Find a candidate best path by Viterbi search
  Step 3: If every score along the path found in Step 2 is a true match score, print the path and stop.
  Step 4: Otherwise, replace every upper bound along the path found in Step 2 with the true match score, and return to Step 2.

- Guaranteed to result in a best path

- Viterbi is performed many times instead of once

Speed Enhancements (Cont.)

Suitable Upper-Bound Heuristics

- Vertical profiles: Count black pixels in each column of the observed and hypothesized images; derive upper bound on match score by assuming they maximally overlap.
- Extension to multilevel templates: In each column, count pixels at each level for the observed and hypothesized images. Obtain upper bound in a greedy fashion, by matching pixels that “care the most” first.
- Bounds can be computed at coarse (i.e., subsampled) resolutions for further speed gains (Bloomberg, 1998)

Speed Enhancements (Cont.)
Incremental Viterbi

- Observation: number of ICP iterations (Viterbi computations) can be high
- Most changes tend to be spatially localized in "problem areas"
- Can we re-use the parts of the path that don't change from the previous iteration, avoiding redoing the full Viterbi computation?
- Look for wide runs of constant-change in node score – these nodes form an "equipotential barrier" that insulates path changes on one side of the barrier from changes on the other.
- ICP and incremental Viterbi in concert were found to improve speed by a factor of about forty.

Extension to Grayscale

- Resolution becomes an important issue
- Edge pixels are crucial
- Edge pixels exhibit systematic variation due to sampling; this is poorly modeled by an independent noise assumption

Extension to Grayscale (Cont.)

- Degraded, subsampled grayscale textline image ($\sigma = 10$):

- Result of Document Image Decoding:

  `WHITE KITTEN HAD BEEN HAVING ITS FACE WASHED BY THE OLD CAT FOR`

- Binarization

- Result of applying industry-standard *TextBridge* program:

  `WHITE LIT EN ,,AS SEEN HAHISE ITS PACE WASH SAT THE ASS CAT EAR`

Extension to Grayscale (Cont.)

- Degraded, subsampled grayscale textline image ($\sigma = 50$):

- Result of Document Image Decoding:

```
WHITE KITTEN HAD BEEN HAVING ITS FACE WASHED BY THE OLD CAT FOR
```

- Degraded, subsampled grayscale textline image ($\sigma = 100$):

- Result of Document Image Decoding:

```
WHITC.KITTEN HAO PFEN HAVINC LI,5.B6GF.59ASHED BY THE,OI.L 3 AT FI?R
```

Imposing Linguistic Constraints on DID

- Goal: to get DID to prefer linguistically valid strings over invalid ones
- Some possible approaches:
  - Run ispell on output
  - Recognize into a lexicon
  - K-best rescoring
  - Direct integration of linguistic constraints into best-path search

Language Modeling

- Language model: a probability distribution over finite strings
- Possibilities for modeling variability of string length:
  - Treat length as an explicit random variable
  - Augment alphabet to contain a termination symbol, and define a tree source.
- Requirements: causality, Markovianity, stationarity, nonzero probability assignment to every finite string
- N-grams tend to be effective

Language Modeling (Cont.)

- Pseudorandom text by sequential sampling from a 4-gram model trained on the Brown corpus:

```
And fine alone other Itality and agains she tumor his result, from
of Brannot faculous shall precentative inter at lear time to much a
relanguages proposal?  Baer fall over Open-megaw of most used agreen
during represearchbishes of chlor] But the first position a little
rear intenant year-olds to And also if we should beef childing face
graduates
```

Language Modeling (Cont.)

- Design and Implementation Issues:
  - Choice of N
  - Mechanisms for varying N
  - Data sparseness: unseen events in known contexts (smoothing)
  - Data sparseness: unknown contexts (backing off)
  - Chain-rule factorization with smoothing gives us only one shot at killing a bad string before we wander into a no-man's land of unknown contexts

Language Model Integration

- Trellis becomes tree-like
  - Complexity determined by N
  - No longer feasible to represent or search over full trellis
- Search techniques:

  A*

  Stack algorithm

  ICP-like approach

Language Model Integration (Cont.)

ICP-like Approach (Popat et al., 2000 (handout))

Step 1: Constuct trellis, labelling each edge with sum of the match score and the unigram (unconditional) upper bound language model score

Step 2: Find a candidate best path by Viterbi search, restricting the search to consider in each spatial position only those nodes that have the most specific context consistent with path history.

Step 3: If every node along the path found in Step 2 is of maximal contextual order with respect to the language model, then output the path and stop.

Step 4: For each node along the found path that is not of maximal contextual order, create a new node in the same spatial location with a context that looks back along the path one additional character, then return to Step 2.

Language Model Integration (Cont.)

- Observations:
  - *List* and K-best algorithms constrain complexity and allow aribtrary language models, but make no guarantee about quality of output
  - *Stack* algorithm allows arbitrary language models, but controlling complexity requires care, and no guarantee is made about the quality of the output
  - A* guarantees the quality of the output, but requires good bounds to be efficient and computation may explode for long lines
  - ICP-like procedure guarantees the quality of the output when it is allowed to terminate on its own. Can also be stopped whenever desired to yield current best path (i.e., it is an “any-time” algorithm).

Language Model Integration Examples

- Degraded, subsampled grayscale textline image ($\sigma = 100$):

- Decoding without a language model:

`WHITC.KITTEN HAO PFEN HAVINC LI,5.B6GF.59ASHED BY THE,OI.L 3 AT FI?R`

- Unigram language model via Viterbi:

`WHITE KITTEN HAO RFEN HAVING ITS EACF WAEHEO,BY THE OLD CAT EOR`

- ICP-like algorithm:

`WHITE KITTEN HAD BEEN HAVING ITS E.BIV WASHED BY THE QLE.3 AT FOR`

- Ground truth:

`WHITE KITTEN HAD BEEN HAVING ITS FACE WASHED BY THE OLD CAT FOR`

Research Challenges

- Template Estimation
  - Aligned, supervised template estimation
  - Estimation based on partially labeled data
  - Use of font libraries
  - Estimation of grayscale templates
- Automatic or semiautomatic inference of Markov source structure
- Using features besides template matches
- Speed and robustness issues
- Extension to color and non-paper-oriented documents