Presented at IAPR 2001 International Conference on Document Analysis and Recognition (ICDAR 2001), Seattle, WA, Sept. 2001

Document Image Decoding using Iterated Complete Path Search with Subsampled Heuristic Scoring

Dan S. Bloomberg
Leptonica
(bloomberg@ieee.org)

Thomas P. Minka
Massachusetts Institute of Technology, Cambridge, MA 02139 USA
(minka@stat.cmu.edu)

Kris Popat
Xerox Palo Alto Research Center, Palo Alto, CA 94304 USA
(popat@ieee.org)

February 12, 2002

Abstract

It has been shown that the computation time of Document Image Decoding can be significantly reduced by employing heuristics in the search for the best decoding of a text line. In the Iterated Complete Path (ICP) method, template matches are performed only along the best path found by dynamic programming on each iteration. When the best path stabilizes, the decoding is optimal and no more template matches need be performed. In this way, only a tiny fraction of potential template matches must be evaluated, and the computation time is typically dominated by the evaluation of the initial heuristic upper-bound for each template at each location in the image.

The time to compute this bound depends on the resolution at which the matching scores are found. At lower resolution, the heuristic computation is reduced, but because a weaker bound is used, the number of Viterbi iterations is increased. We present the optimal (lowest upper-bound) heuristic for any degree of subsampling of multilevel template and/or interpolation, for use in text line decoding with ICP. The optimal degree of subsampling depends on image quality, but it is typically found that a small amount of template subsampling is effective in reducing the overall decoding time.

Keywords: document image decoding, dynamic programming, Viterbi, heuristic search, optical character recognition, iterated complete path, MAP, template matching, subsampling

1 Introduction to DID with ICP

In the *Document Image Decoding* (DID) approach of Kopec and Chou [1], a model of the generation process is combined with a noise model to allow determination of a *maximum a posteriori* (MAP) message (decoding) given an observed image. The model of the generation process includes templates for the possible printed characters, a language model specifying a prior on the templates or sequences of templates, a setwidth parameter for each template describing its width, and a grammar that describes the syntax of sequences of printed characters. The simplest grammar is a finite state machine for a line of text, specified by a start node, an end node, and a printing node that loops back to itself as each character is printed. In this introduction, we give a quick review of the fundamental parameters and equations for a DID line decoder, and then describe the *Iterated Complete Path* (ICP) algorithm. In Section 2 we describe simple heuristic bounds for full-resolution and for subsampled convolution, both without and with interpolation. Then in Section 3 we show the computational efficiency for ICP line decoding as a function of subsampling and interpolation parameters.

1.1 DID with Multilevel templates

Kopec [5] used multilevel character templates Q, where the pixels in each template are assigned to one of several (disjoint) sub-templates $Q^{(l)}$. Level $l = 0$ is the background; α_0 is the probability that a background pixel is OFF in the observed image. For the sub-templates $Q^{(l)}, l > 0$, the parameters α_l give the probability that a pixel is ON. For a *write-black* sub-template, a pixel is more likely to be ON than for the background; hence, $\alpha_l > (1 - \alpha_0)$. In a *write-white* subtemplate, $\alpha_l < (1 - \alpha_0)$. For an L-level template, the $L - 1$ non-background levels contribute to the *log normalized likelihood* of observing an image Z when a template Q is printed:

$$\begin{aligned} \mathcal{L}(Z \mid Q) &= \sum_{l=1}^{L-1} \mathcal{L}(Z \mid Q^{(l)}) && (1) \\ &= \sum_{l=1}^{L-1} \left[\gamma_l \, \|Q^{(l)} \wedge Z\| \; + \; \beta_l \, \|Q^{(l)}\| \right] && (2) \end{aligned}$$

where $\|X\|$ denotes the number of 1's in X, $\wedge$ is the bitwise **and** operator, and for levels $l > 0$

$$\begin{aligned} \gamma_l &= \log \frac{\alpha_0 \alpha_l}{(1 - \alpha_0)(1 - \alpha_l)} && (3) \\ \beta_l &= \log \frac{1 - \alpha_l}{\alpha_0} && (4) \end{aligned}$$

Note that the *background* level is treated as *don't care*; the write-black levels l have $\gamma_l > 0$ and $\beta_l < 0$; and the write-white levels have $\gamma_l < 0$ and $\beta_l > 0$. For write-black sub-templates, $\mathcal{L}(Z \mid Q^{(l)})$ increases with the number of black pixels that fall under $Q^{(l)}$, whereas for write-white sub-templates, $\mathcal{L}(Z \mid Q^{(l)})$ is maximal when no black pixels fall under $Q^{(l)}$.

```
initialize score array with upper-bound heuristics
$i \leftarrow 0$, score(0) $\leftarrow 0$
do
    $i \leftarrow i + 1$
    find best path using Viterbi on current score array
    score($i$) $\leftarrow$ score of that best path
    rescore array along best path
until: score($i$) = score($i - 1$)
```

Figure 1: Algorithm for ICP with upper-bound heuristic

Kopec and Chou [1] showed that with a simple Markov source, corresponding to a unigram language model where the priors on the templates are independent of history, a MAP path for the line decoder is found by solving the recursive equation for the MAP function $\mathcal{F}$ at successive pixel locations x as a function of the likelihoods at previous pixel positions. With the prior transition probability a_t for a transition t that includes a matching score for a template Q_t accompanied by a translation Δ_t,

$$\mathcal{F}(x) = \max_{t \mid R_t = x} \{\mathcal{F}(x - \Delta_t) + \log a_t + \mathcal{L}(Z \mid Q_t[x - \Delta_t])\} \tag{5}$$

where the maximum is taken over all transitions originating from the template origin at $x - \Delta_t$ and whose right side R_t is located at x. The computation of the multilevel template match scores $\mathcal{L}(Z \mid Q_t[x])$ is found from (2).

We assume that as a pre-processing step, the baselines for each textline have been identified, as in Chen et al., [3], and are typically within one pixel of the correct vertical location at all points along the text line.

1.2 DID with ICP heuristics

The ICP method was first used in DID by Kam and Kopec [2, 4]. A heuristic upper bound on the score from a complete line was used to reduce the computation for separable models, where a 2D page image can be decoded as a sequence of 1D line decoders. The heuristic allowed them to locate the approximate position of each text line, where full line decodings were then performed. Recently, the ICP method has been used in DID to reduce the decoding computation of each line [6]. Given an upper-bound heuristic, the ICP algorithm for line decoding is as shown in Figure 1.

In the rescoring operation, the actual template match score is found at typically five vertical positions around the nominal baseline, and the largest score is inserted in the score array. Also, to reduce the number of iterations, the template is also matched in the vicinity of the node, typically for one pixel on each side, and the rescored values are inserted in the score array. Otherwise, a new path is likely to be found with the same template in a neighboring x location, because it is using the (usually larger)

```
set skip-mode ON
for each $x$ from 0 to $w$
{
    if skip-mode is ON,
        if we hit a rescored node,
            set skip-mode OFF
            Icount $\leftarrow 0$

    otherwise if skip-mode is OFF
        if $\Delta_{\mathcal{F}}(x) = \Delta_{\mathcal{F}}(x-1)$,
            Icount $\leftarrow$ Icount $+1$
            if Icount $> \Delta_{max}$
                set skip-mode ON
        else
            Icount $\leftarrow 0$ (reset)
}
```

Figure 2: *Algorithm for Incremental Viterbi*

heuristic value in evaluating paths going through the neighboring nodes. It is also important to mark which nodes in the array have been rescored, so that the same node is not rescored more than once.

1.3 DID with Incremental Viterbi

With ICP, and the reduction of the computation for the score array, the Viterbi iterations become a significant factor in the overall computation time, because for M templates, a comparison is required of M scores at each pixel location x in the forward direction. As iterations proceed, and convergence is approached, changes become highly localized, and much of the best path remains invariant. It is shown in [6] that it is possible to identify those parts of the *best* path that will not differ in the current iteration from the previous one, and for such regions, the forward best-path search can be placed in *skip-mode*, where the best transitions from the previous iteration are retained, and the partial path score is updated with an additive constant, referred to below as $\Delta_{\mathcal{F}}$. The criterion for entering *skip-mode* is that the difference in partial scores between the previous and current iterations, $\Delta_{\mathcal{F}}$, doesn't change for a distance equal to the largest character setwidth, Δ_{max}. When a rescored node is encountered, we must leave *skip-mode* and enter *slow-mode*. Each text line can begin in *skip-mode*, and makes a transition to *slow-mode* when the first rescored node is reached. Use of incremental Viterbi typically reduces the Viterbi computation by about a factor of two. The control logic for incremental Viterbi is shown in Figure 2.

2 Heuristic Upper Bounds

To guarantee that a MAP path will be found, a strict upper-bound for the matching score must be used. In this section, we give the minimum upper-bound scores that can be derived for one-dimensional arrays consisting of column pixel sums. Only the image-dependent term in (2) is given here; the template term is evaluated exactly. In the following, we use I_j to represent the sum of foreground pixels in the j^{th} column of the image, and $\vec{T}_j$ to represent the *array* of pixel sums in the j^{th} column of *each subtemplate* T of template $\vec{T}$. For bilevel templates, $\vec{T}_j$ is the single sum of foreground pixels T_j for that template.

In section 2.1 we give the column-based heuristic upper-bound at full resolution. Then in section 2.2 we perform subsampled upper-bound convolutions for templates at every location $x = 0, 1, ...$ along the text line. This is called a Template Subsampling, or *T-subsampling*. Finally, in section 2.3 we show that the convolution can be performed at every m^{th} location $x = 0, m, 2m, ...$, and that an upper-bound heuristic can be interpolated between these points. This is called an Interpolation Subsampling, or *I-subsampling*. If a subsample factor n is used in the T-subsampling and a subsample factor m is used in the I-subsampling, the computation of the heuristic upper-bound is reduced by approximately nm.

2.1 Full resolution heuristic scores

Place the origin of template $\vec{T}$ at location $x = i$. Then the image-dependent part of the heuristic score for the j^{th} column of multilevel template $\vec{T}$ is given by a filling function f:

$$H_j(i) = f(I_{i+j}, \vec{T}_j) \tag{6}$$

For multilevel templates, the function f fills the subtemplates with I_{i+j} pixels, starting with the write-black level k with largest γ_k. Pixels are placed in write-white levels (with negative γ_k) only after both the write-black and background (don't-care) levels have been filled.

For bilevel templates, the contribution from the j^{th} column to the heuristic score is simply

$$H_j(i) = \gamma_1 \min(I_{i+j}, T_j) \tag{7}$$

Then the heuristic score for a template with origin at at $x = i$ is given by the one-dimensional convolution

$$H(i) = \sum_{j=0}^{j=w-1} H_j(i) \tag{8}$$

2.2 Subsampled template convolution heuristic scores

We first consider T-subsampling. By filtering and subsampling image and column sums, the number of operations in the convolution is reduced by the subsampling factor. The full resolution heuristic score H is then replaced by a subsampled heuristic score S^n that bounds it above. Assume a subsampling factor $n = 2$. The filter that gives the smallest upper-bound heuristic score is the min. This is easily seen for bilevel templates. Consider the contribution to H from two adjacent columns:

$$H_j(i) + H_{j+1}(i) = \gamma_1(\min(I_{i+j}, T_j) + \min(I_{i+j+1}, T_{j+1})) \tag{9}$$

Because the min of sums is greater than or equal to the sum of mins, an upper-bound for the two-column heuristic is

$$U_j^2(i) = \gamma_1 \min(I_{i+j} + I_{i+j+1}, T_j + T_{j+1}) \geq H_j(i) + H_{j+1}(i) \tag{10}$$

Store the adjacent image sums in a full resolution array

$$I_i^2 \equiv I_i + I_{i+1}, i = 0, 1, ... \tag{11}$$

and the template sums for a template of width w in a half-resolution array

$$T_j^2 \equiv T_j + T_{j+1}, j = 0, 2, ..., w-2 \tag{12}$$

where the superscript 2 indicates the sum over two adjacent columns.
The upper-bound for the two-column heuristic is

$$U_j^2(i) = \gamma_1 \min(I_{i+j}^2, T_j^2) \tag{13}$$

and the subsampled heuristic score is then

$$S^2(i) = \sum_{j=0,2,...}^{w-2} U_j^2(i) = \sum_{j=0,2,...}^{w-2} \gamma_1 \min(I_{i+j}^2, T_j^2) \tag{14}$$

More generally, for *multilevel* templates the upper-bound for the two column heuristic is

$$\mathcal{U}_j^2(i) = f(I_{i+j}^2, \vec{T}_j^2) \tag{15}$$

where the filling function f is evaluated as before, except using pairwise sums of image and template column pixels. The subsampled heuristic score for a template $\vec{T}^2$ at $x = i$ is

$$S^2(i) = \sum_{j=0,2,...}^{w-2} \mathcal{U}_j^2(i) = \sum_{j=0,2,...}^{w-2} f(I_{i+j}^2, \vec{T}_j^2) \tag{16}$$

This sum is performed for each location $x = i$, so the reduction in computation is proportional to the subsample factor n. The extension to subsampled convolution $S^n(i)$ by a factor $n > 2$ is obvious.

2.3 Subsampled convolution with interpolation

With both T- and I-subsampling, the full resolution heuristic is replaced by a subsampled heuristic score $S^{n,m}$ that bounds it from above. For simplicity, consider an I-subsampling with a factor $m = 2$, in addition to a T-subsampling with factor $n = 2$. We wish to find the contribution from two columns j and $j+1$ at both locations $i+j$ and $i+j+1$, using only a single table lookup. For *bilevel* templates, using (13), and interchanging the min and max operations, we have

$$U_j^{2,2}(i) \equiv \max(U_j^2(i), U_j^2(i+1)) \tag{17}$$
$$= \gamma_1 \min(\max(I_{i+j}^2, I_{i+j+1}^2), T_j^2) \tag{18}$$

Defining a subsampled array

$$I_{i+j}^{2,2} \equiv \max(I_{i+j}^2, I_{i+j+1}^2), i = 0, 2, ... \tag{19}$$

the doubly subsampled upper-bound heuristic score is then

$$S^{2,2}(i) = \sum_{j=0,2,...}^{w-2} U_j^{2,2}(i) \tag{20}$$
$$= \sum_{j=0,2,...}^{w-2} \gamma_1 \min(I_{i+j}^{2,2}, T_j^2), i = 0, 2, ... \tag{21}$$

The score $S^{2,2}(i)$ is evaluated at even locations for $x = i$, and the value is replicated to the odd locations at $x = i - 1$. Hence the computation is reduced by 4 compared with the full resolution heuristic score. Extension to higher I-subsampling $S^{2,m}$ is obvious.

More generally, for *multilevel* templates where all levels are write-black ($\gamma > 0$), the upper-bound for the two-column heuristic with $m = 2$ subsampling is

$$\mathcal{U}_j^{2,2}(i) = f(I_{i+j}^{2,2}, \vec{T}_j^2) \tag{22}$$

and the subsampled heuristic score is

$$S^{2,2}(i) = \sum_{j=0,2,...}^{w-2} \mathcal{U}_j^{2,2}(i) \tag{23}$$
$$= \sum_{j=0,2,...}^{w-2} f(I_{i+j}^{2,2}, \vec{T}_j^2), i = 0, 2, ... \tag{24}$$

Again, these scores are replicated to odd i, and the extension to I-subsampling with a factor $m > 2$ is obvious. However, if there exists a write-white level, I-subsampling is not possible because the maximum score is not necessarily given by the maximum of the sum of pixels in (19). For example, with a space character that has *only* write-white pixels, the upper-bound would be found by *minimizing*, rather than maximizing, the counts.

3 Discussion

In standard DID, without ICP, a matrix of exact matching scores is computed for all templates at all positions. For each pixel position across the text line, matching scores are performed with 32-bit aligned operations between each of the (32-bit aligned) templates and a buffer into which a fragment of the image starting at that pixel position has been placed. After each pair of 32-bit words is ANDed, table lookup is used to determine the number of ON pixels. The results are weighted for each sub-template by the appropriate γ factor for that level.

It has been previously described how the contribution of a column of template pixels to the heuristic score for that template at any location can be found from a single table lookup, regardless of the number of levels in the template [6]. In brief, for each template, a two-dimensional lookup table is generated that gives the heuristic score for each column in a multilevel template as a function of the number of ON pixels that may be placed in the column. Once generated, this table is used with the actual number of pixels in the image column (or columns, depending on the degree of subsampling) to find the contribution to the heuristic template score at that location. Whereas the computation of the exact matching score for a template is proportional to the number of sub-templates, the computation of the heuristic score by lookup table is independent of the number of sub-templates.

As in [6], we use *incremental Viterbi*, where the forward Viterbi trellis is pruned to follow the result in the previous iteration (*skip-mode*) whenever possible. When not in skip-mode, the possibility that each template can be placed at each location is computed; this is called *slow-mode*. We also rescore two nodes adjacent to those found for the best path, as this was found to significantly reduce the number of Viterbi iterations with negligible computational overhead, and rescoring more than two nodes gives little improvement. With this method, ICP line decoding is about 30x faster than computing exact matching scores.

With both T and I subsampling, the contribution of n columns of the template to an upper-bound heuristic at any of m different adjacent locations can also be found by a single table lookup, giving an approximate reduction in heuristic computation by a factor of nm. Of course, the subsampled bound is less tight, and more Viterbi iterations are required for convergence. Here we show the trade-off between heuristic computation and Viterbi iterations.

Table 1 shows the computation required for decoding a typical line of text using DID with ICP, for different template subsampling factors n. The text line has 85 characters, including white space, and a width of 1960 pixels. The model has 329 4-level templates, of which one of the three subtemplates is write-white. An exact template match then requires 3 bilevel matches, and the node score is taken for the best vertical alignment among 5 vertical positions of the template near the baseline; approximately 60 nodes can be rescored in 1 msec. All times are on a program built with the GNU C compiler, running on a 600 MHz Pentium III.
Without template subsampling, heuristic scoring takes about twice as long as Viterbi. With $n = 2$ subsampling, the heuristic score time is reduced, the number of iterations and rescored nodes increases, the balance between heuristic and Viterbi computation is improved, and the overall decoding time is decreased. For $n > 2$, the increase in the number of Viterbi iterations cancels the further improvement in heuristic computation, and the overall decoding time increases.

In Tables 2 and 3, we show the computation required for decoding the same line of text with ICP using a 3-level template, composed of the two write-black subtemplates in the 4-level template used above. The pixels in the write-white template are put into the background as “don’t care”. The performance is given for combinations (n, m) of template subsampling n and score interpolation m factors, where $m = 1$ in Table 2 and $m = 2$ in Table 3.
Comparing the results in Table 1 with those in Table 2, we see that without the write-white subtemplate,

	Template subsampling			
	1	2	3	4
Rescored nodes	1144	1961	3266	6391
Viterbi iterations	10	15	26	35
Fraction in skip mode	0.43	0.37	0.39	0.24
Fraction in slow mode	0.57	0.63	0.61	0.76
Heuristic score time (sec)	0.62	0.35	0.27	0.23
Rescore node time (sec)	0.02	0.03	0.05	0.10
Viterbi time (sec)	0.29	0.46	0.79	1.31
Rescore + Viterbi time	0.31	0.49	0.84	1.41
Total time (sec)	0.93	0.84	1.11	1.64

Table 1: *Dependence of heuristic score and Viterbi computation on the template subsampling factor, for a 4-level template with two write-black and one write-white levels.*

	T and I factors (n,m)		
	(1,1)	(2,1)	(3,1)
Rescored nodes	3079	4533	6709
Viterbi iterations	18	25	36
Fraction in skip mode	0.36	0.34	0.31
Fraction in slow mode	0.64	0.66	0.69
Heuristic score time (sec)	0.61	0.35	0.26
Rescore node time (sec)	0.05	0.07	0.11
Viterbi time (sec)	0.58	0.83	1.22
Rescore + Viterbi time	0.63	0.90	1.33
Total time (sec)	1.24	1.25	1.61

Table 2: *Dependence of heuristic score and Viterbi computation on both template and interpolation subsampling, for a 3-level template with two write-black levels. This is given for pairs (n = template subsampling, m = 1 = score interpolation).*

	T and I factors (n,m)		
	(1,2)	(2,2)	(3,2)
Rescored nodes	13720	17718	22678
Viterbi iterations	80	105	137
Fraction in skip mode	0.22	0.21	0.21
Fraction in slow mode	0.78	0.79	0.79
Heuristic score time (sec)	0.33	0.20	0.16
Rescore node time (sec)	0.23	0.29	0.33
Viterbi time (sec)	2.96	3.92	5.11
Rescore + Viterbi time	3.19	4.25	5.44
Total time (sec)	3.42	4.45	5.60

Table 3: *Dependence of heuristic score and Viterbi computation on both template and interpolation subsampling, for a 3-level template with two write-black levels. This is given for pairs (n = template subsampling, m = 2 = score interpolation).*

the heuristic upper bound is poorer, and the number of Viterbi iterations is larger. Further, with the 3-level template, T-subsampling for $n = 2$ produces no appreciable improvement. Most importantly, Table 3 shows that the upper-bound with I-subsampling is very poor, and causes a significant increase in the number of Viterbi iterations to achieve convergence. Furthermore, those iterations spend a greater fraction in slow mode, again indicating that the best path during most of these iterations is far from the true MAP path. (Of course, at convergence the same MAP decoding is ultimately found, because all heuristics are true upper-bounds.)

4 Conclusions

Using ICP without template or interpolation subsampling and for a typical line of text, about two-thirds of the computation is in the heuristic scoring, about one-third is in the dynamic programming, and the decoding time for a 2000 pixel wide column of text using 4-level templates is about 2.5 msec/template. T-subsampling by $n = 2$ gives a small improvement in the efficiency of a line decoder, over the non-subsampled method. The heuristic upper-bounds are much tighter for multilevel templates that include at least one write-white level; consequently, the number of Viterbi iterations is significantly reduced for such templates over those with only write-black subtemplates. This factor mitigates against using I-subsampling, because there is no useful upper-bound heuristic for multilevel templates with write-white subtemplates. Furthermore, I-subsampling is not practical, because of the great increase in Viterbi iterations that results from a significantly looser upper-bound in the heuristic.

References

[1] G. Kopec and P. Chou, “Document image decoding using Markov source models”, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, June, 1994, pp. 602–617.

[2] A. Kam and G. Kopec, “Separable source models for document image decoding”, in *Document Recognition II*, L. Vincent and H. Baird, editors, Proc. SPIE vol. 2422, pp. 84–97, 1995.

[3] F. Chen, D. Bloomberg and L. Wilcox, “Spotting phrases in lines of imaged text”, *Document Recognition II*, L. Vincent and H. Baird, editors, Proc. SPIE vol. 2422, pp. 256–269, 1995.

[4] A. Kam and G. Kopec, “Document image decoding by heuristic search,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 9, Sept. 1996, pp. 945-950.

[5] G. Kopec, “Multilevel character templates for document image decoding” in *Document Recognition IV*, L. Vincent and J. Hull, editors, Proc. SPIE vol. 3027, pp. 168–179, 1997.

[6] T. Minka, D. Bloomberg and K. Popat, “Document image decoding using iterated complete path search” in *Document Recognition and Retrieval VIII*, P. Kantor, D. Lopresti and J. Zhou, editors, Proc. SPIE vol. 4307, pp. 250–258, 2001.