user drawing. It is noteworthy that the proposed approach is also applicable to character recognition. Fig. 10 shows an example of recognition of English characters printed on pictorial backgrounds. In a great number of experiments, the desired image appeared in the first three among 83% of queries, and in the first eight among 96% of queries.

There are two reasons the retrieval may fail. First, the desired image is too complicated to paint in a query picture. If the desired image has diversified colors or texturelike complex patterns, it is difficult for users to specify or paint it in a query. Second, when the area of "not-cared" regions is quite large, they would become too dominant to match the desired images.

Because of the dynamic programming nature of 2-D Viterbi algorithms, the processing involved in query comparison is quite expensive. The time required to process a query in the experiments is about 14–15 min on 200 images. One way to reduce the computational cost is to use a coarse-to-fine matching method. In the method, 64 2-D PHMM's that best satisfy the query using color histogram matching [6] are first obtained. Then, the query picture is rectangularly sampled to a 16×16 image and the 2-D Viterbi algorithm is performed to retrieve the best 32 2-D PHMM's from the 64 2-D PHMM's. Next, the query picture is finely sampled to a 32×32 image and the 2-D Viterbi algorithm is performed again to select the best 16 2-D PHMM's from the 32 2-D PHMM's. The process is iteratively performed until the desired number of 2-D PHMM's are obtained. In the experiments, the time required to retrieve eight images that best match the query is reduced to about 42 s.

To examine the effect of using different color spaces, the luminosity–in-phase–quadrature (YIQ) space is used for comparison in our experiment. While the segmentation based on the RGB and YIQ spaces may appear different, the retrieval performance is similar. As for the effect of different segmentation schemes, the block-based segmentation method is replaced by the c-means algorithm [7]. It is found that the query comparison cost based on the c-means algorithm is much more than that based on block-based segmentation. Moreover, the c-means algorithm usually results in many scattered small regions after segmentation. Since in general a query picture is composed of coherent regions (see Fig. 5), the c-means algorithm is not preferred.

IV. Conclusions and Further Research

A new approach to retrieving images from a color image database based on 2-D PHMM's is proposed. From the experiments, 2-D PHMM's are shown to be flexible in describing the chromatic and spatial information of the images. The additional use of a pictorial querying method and the 2-D PHMM's results in effective query matching. The desired images that partially or exactly satisfy the query picture can be retrieved with high accuracy. Further research may be directed to the following topics: i) Adapting HMM's to descriptions of textures and designing the corresponding querying method by incorporating the syntactic approach; and ii) Improving the performance of the proposed approach by parallelizing and hardware implementation of the critical section to achieve real-time image retrieval.

References

[1] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE,* vol. 77, pp. 257–286, 1989.

[2] O. E. Agazzi and S. S. Kuo, "Hidden Markov model based optical character recognition in the presence of deterministic transformations," *Pattern Recognit.,* vol. 26, pp. 1813–1826, 1993.

[3] R. Duda and P. Hart, *Pattern Classification and Scene Analysis.* New York: Wiley, 1973.

[4] A. P. Dempster *et al.,* "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Stat. Soc.,* vol. 39, pp. 1–38, 1977.

[5] G. D. Forney, "The Viterbi algorithm," *Proc. IEEE,* vol. 61, pp. 268–278, 1973.

[6] M. J. Swain and D. H. Ballard, "Color indexing," *Int. J. Comput. Vis.,* vol. 7, pp. 11–32, 1991.

[7] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms.* New York: Plenum, 1981.

Deinterlacing by Successive Approximation

Jelena Kovačević, Robert J. Safranek, and Edmund M. Yeh

***Abstract*— We propose an algorithm for deinterlacing of interlaced video sequences. It successively builds approximations to the deinterlaced sequence by weighting various interpolation methods. A particular example given here uses four interpolation methods, weighted according to the errors each one introduces. Due to weighting, it is an adaptive algorithm. It is also time-recursive, since motion-compensated part uses the previously interpolated frame. Furthermore, bidirectional motion estimation and compensation allow for better performance in case of scene changes and covering/uncovering of objects. Experiments are run both on "real-world" and computer generated sequences. Finally, subjective testing is performed to evaluate the quality of the algorithm.**

I. Introduction

Interlaced video has been around for quite some time, and along the way many of the problems associated with it have been discovered, such as line crawl and interline flicker. Moreover, interlaced video makes motion-based processing very difficult, resampling is hard, and, furthermore, it makes little sense displaying a single frame out of a sequence. For these and other reasons, the current trend is toward progressively scanned video. However, although there exist very good interlaced cameras due to physical limitations in the current technology, progressive cameras are still a topic of ongoing research. Recently, the US HDTV Grand alliance has put forward a proposal containing both interlaced and progressive scanning formats. Therefore, there exists a need for a good deinterlacing system. Also, with the integration of multimedia services on desktops and workstations, one needs progressive video for display.

Previous work on deinterlacing can be roughly divided into four categories: In the first are methods using purely spatial techniques as in [1]; the second group consists of those methods based on median filtering, both in two and three dimensions as in [2]; the third group employs motion adaptive techniques as in [3]; and the fourth group comprises methods that use motion compensation as in [4]. Recently, in [5], the authors presented a framework for estimating two-dimensional (2-D) motion fields from image sequences, which

Manuscript received April 20, 1995; revised June 6, 1996.

J. Kovačević and R. J. Safranek are with Bell Laboratories, Innovations for Lucent Technologies, Murray Hill, NJ 07974 USA (e-mail: jelena@research.bell-labs.com).

E. M. Yeh is with the Department of Electrical Engineering, The Massachusetts Institute of Technology, Cambridge, MA 02139 USA.

Publisher Item Identifier S 1057-7149(97)00340-0.

is general, and incorporates object acceleration, nonlinear motion trajectories and occlusion effects. Computed motion fields are then applied to various tasks in image sequence processing, one of which is deinterlacing [6]. Our algorithm could be seen as a generalization of what Nguyen and Dubois did in [6].

The idea behind the algorithm is to successively refine our deinterlaced frame by first employing interpolation methods that use only existing pixels. These methods are weighted to produce the first approximation to our deinterlaced frame. At each subsequent stage, we employ interpolation methods that need higher and higher degrees of accuracy of interpolation. We weight each new method with the already existing ones to produce a further approximation to the deinterlaced frame. Typically, the final interpolation method to be used is based on motion compensation, since it needs to be very accurate in order to produce good results. A specific instance of what we just described is based on the spatiotemporal interpolation algorithm of [6]. Experiments are run both on "real-world" and computer-generated sequences.

II. Deinterlacing Algorithm

Let us start with a general formulation of the problem as given in [5]. The idea is to estimate 2-D motion fields from a sequence and then perform interpolation along the estimated motion trajectory. We want to interpolate the missing lines. Thus, let us try to predict the value of the pixel at position (h, v, t), where h, v, t stand for horizontal, vertical and time dimensions, respectively. In what follows $I(h, v, t)$ will denote the intensity of the pixel at location (h, v, t), where the pixel is taken from the original field. Since at a given time t, we have some input image samples (unlike when no samples are available, as in field rate standards conversion), it makes sense to combine spatial with temporal interpolation as follows:

$$I^*(h, v, t) \;=\; \xi I_T^*(h, v, t) \;+\; (1 - \xi) I_S^*(h, v, t).$$

Here, I_S^* refers to the result of spatial interpolation, while I_T^* refers to motion-compensated temporal interpolation, along the estimated motion trajectory. In what follows, I_S^* and I_T^* are split even further; I_S^* will consist of vertical and orientational interpolation, while I_T^* will comprise purely linear temporal as well as motion-compensated interpolation.

The idea behind the algorithm is to successively refine our deinterlaced frame by first employing interpolation methods $I_{m,i}, i = 1, \cdots, M$ that use only existing pixels. These methods are weighted to produce the first approximation to our deinterlaced frame as

$$I_1 = \sum_{i=1}^{M} k_i I_{m,i}.$$

At each subsequent stage $j = 2, \cdots$, we employ interpolation methods $I_{m,M-1+j}$ that need higher and higher degrees of accuracy of interpolation and which, as input, have the previous approximations. We weight each new method with the already existing ones to produce a further approximation to the deinterlaced frame as

$$I_j = \sum_{i=1}^{M+j-1} k_{i_j} I_{m,i}.$$

Typically, the final interpolation method to be used is based on motion compensation, since it needs to be very accurate in order to produce good results. A specific instance of what we just described is based on the spatiotemporal interpolation algorithm of Nguyen and Dubois [6]. We use their four interpolation methods in an embedded way, as will be shown next.

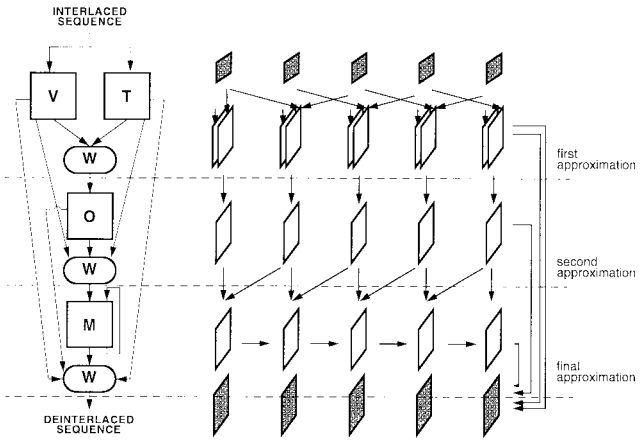


Fig. 1. Structure of the algorithm. The figure shows how the sequence is built up through successively approximating the values to be interpolated. The first approximation is obtained by weighting vertical and temporal blocks, and is subsequently used to obtain the orientational block. Then vertical, temporal and orientational methods are weighted to obtain the second approximation used for motion compensation. The final step is to weight all four methods producing the final deinterlaced sequence.

1) First Level of Approximation: We will start by using two methods employed in [6] that need only existing pixels, that is, vertical interpolation I_v and temporal interpolation I_t. For vertical interpolation we say that the intensity at location (h, v, t) is

$$I_v(h, v, t) \;=\; \frac{1}{2}(I(h, v-1, t) + I(h, v+1, t)).$$

Since we will need it later, we compute here the neighborhood difference of vertical interpolation

$$D_v \;=\; \sum_{i=h-2}^{h+2} |I(i, v-1, t) - I(i, v+1, t)|. \tag{1}$$

On the other hand, temporally interpolated values are obtained as follows:

$$I_t(h, v, t) \;=\; \frac{1}{2}(I(h, v, t-1) + I(h, v, t+1)).$$

We now introduce the weighting of these two methods. I_1 will denote the interpolated value at location (h, v, t), using vertical and temporal interpolation and represents the first approximation to the deinterlaced frame. Thus

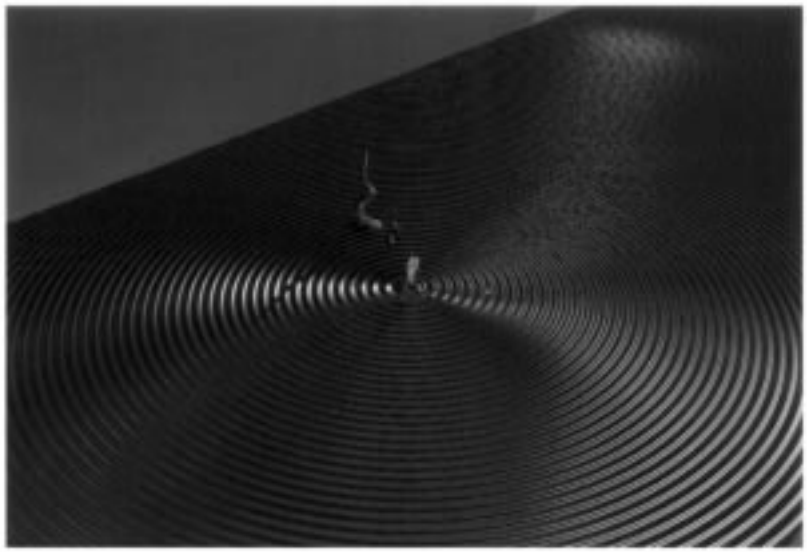
$$I_1(h, v, t) \;=\; \hat{k}_v I_v(h, v, t) + \hat{k}_t I_t(h, v, t)$$

where the weighting factors are computed from the neighborhood difference D_v given in (1) and neighborhood difference D_t as[1]

$$\begin{aligned} D_t \;&=\; \sum_{i=h-2}^{h+2} |I(i, v, t-1) - I(i, v, t+1)| \\ \hat{k}_v \;&=\; \frac{D_t}{D_v + D_t} \\ \hat{k}_t \;&=\; \frac{D_v}{D_v + D_t}. \end{aligned}$$

[1]Note that here $\hat{k}_v$ and $\hat{k}_t$ correspond to k_1 and k_2, while I_v and I_t correspond to $I_{m,1}$ and $I_{m,2}$.

(a)



(b)

(c)

Fig. 2. A frame from each of the sequences used in the experiments. (a) Megacycles. (b) Fish. (c) Raft.

TABLE I
RESULTS OF THE SUBJECTIVE TEST: COMPARISON BETWEEN THIS ALGORITHM (US) AND THE REFERENCE ONE (HD). ND STANDS FOR "NO DIFFERENCE," WHILE CD STANDS FOR "CANNOT DECIDE." FIRST COLUMN UNDER EACH SEQUENCE NAME REPRESENTS THE NUMBER OF PEOPLE WHO CHOSE THAT OPTION; SECOND COLUMN IS THE RELATIVE SCORE

Us vs. Hd	mega		fish		raft	
Us	9	23/30	7	14/30	1	2/30
Hd	1	2/30	1	1/30	3	4/30
ND	0		0		5	
CD	0		2		1	

TABLE II
RESULTS OF THE SUBJECTIVE TEST: COMPARISON BETWEEN THIS ALGORITHM (US) AND THE ORIGINAL (OR). ND STANDS FOR "NO DIFFERENCE," WHILE CD STANDS FOR "CANNOT DECIDE." FIRST COLUMN UNDER EACH SEQUENCE NAME REPRESENTS THE NUMBER OF PEOPLE WHO CHOSE THAT OPTION; SECOND COLUMN IS THE RELATIVE SCORE

Us vs. Or	mega		fish		raft	
Us	1	1/30	1.5	2.5/30	0	0
Or	7	15/30	7.5	16.5/30	9	15/30
ND	2		0		0	
CD	0		1		0	

2) Second Level of Approximation: We now try to build the second level of approximation by adding another method from [6], that is, orientational interpolation. It estimates the local edge orientation according to [7], the result of which is the angle $\beta_{(h,v)}$. It is obtained by using steerable filters, which can have any orientation and are synthesized as a linear combination of certain basis filters. Here, as in [7], we use the second derivative of the Gaussian filter as the basis function at G^{0°, and select another two basis filters G^{60°, G^{120°. To synthesize phase, a filter approximating the Hilbert transform is used [7] together with another three basis filters. To estimate the edge orientation, we use the field already deinterlaced using the above two methods (vertical and temporal); that is, the first approximation I_1. Therefore

$$I_o(h,v,t) = \frac{1}{2}(I_1(h-\delta_h, v-1, t) + I_1(h+\delta_h, v+1, t))$$

where $\delta_h = 0.5/\tan(\beta_{(h,v)})$ (for the luminance channel). The neighborhood difference of orientational interpolation is

$$D_o = \sum_{i=h-2}^{h+2} |I_1(i-\delta_h, v-1, t) - I_1(i+\delta_h, v+1, t)|.$$

Then, the interpolated value obtained using these three methods I_2, that is, the second approximation to the deinterlaced frame, is obtained as[2]

$$I_2(h,v,t) = \tilde{k}_v I_v(h,v,t) + \tilde{k}_t I_t(h,v,t) + \tilde{k}_o I_o(h,v,t)$$

with

$$\tilde{k}_v = \frac{D_t D_o}{D_v D_t + D_v D_o + D_t D_o}$$
$$\tilde{k}_t = \frac{D_v D_o}{D_v D_t + D_v D_o + D_t D_o}$$
$$\tilde{k}_o = \frac{D_v D_t}{D_v D_t + D_v D_o + D_t D_o}.$$

[2]Note that here $\tilde{k}_v$, $\tilde{k}_t$ and $\tilde{k}_o$ correspond to k_{1_2}, k_{2_2}, and k_{3_2}, respectively, while I_o corresponds to $I_{m,3}$.

TABLE III
RESULTS OF THE SUBJECTIVE TEST: COMPARISON BETWEEN THE REFERENCE ALGORITHM (HD) AND THE ORIGINAL (OR). ND STANDS FOR "NO DIFFERENCE," WHILE CD STANDS FOR "CANNOT DECIDE." FIRST COLUMN UNDER EACH SEQUENCE NAME REPRESENTS THE NUMBER OF PEOPLE WHO CHOSE THAT OPTION; SECOND COLUMN IS THE RELATIVE SCORE

Hd vs. Or	mega		fish		raft	
Hd	0	0	0.5	0.5/30	1	1/30
Or	10	26/30	9.5	22.5/30	9	19/30
ND	0		0		0	
CD	0		0		0	

3) Final Approximation: To build our final version of the deinterlaced frame, we add another method from [6]: motion compensation. Here, we first estimate motion and then interpolate along the direction of motion. Motion estimation is performed by block matching in a given search window. Since we need very accurate estimates, we will tend toward smaller blocks and large search areas. If the estimates are not accurate enough, the algorithm will not give adequate weight to the motion compensation part. We will interpolate bidirectionally; that is, we will use both forward and backward motion estimation. Also, the interpolation is performed recursively [8], since backward interpolation will use a frame that has already been deinterlaced using all four methods, that is, the final approximation, denoted by I_3. On the other hand, forward interpolation will use a frame that has been deinterlaced using only the first three methods; that is, the second approximation I_2. Therefore

$$I_m(h,v,t) = k_b I_3(h-d_{h_b}, v-d_{v_b}, t-1) + k_f I_2(h-d_{h_f}, v-d_{v_f}, t+1)$$

where (d_{h_b}, d_{v_b}) and (d_{h_f}, d_{v_f}) are the backward and forward displacement vectors, respectively. The neighborhood differences of motion-compensated interpolation are

$$D_{m_b} = \sum_{i=h-2}^{h+2} |I_2(i,v,t) - I_3(i-d_{h_b}, v-d_{v_b}, t-1)| \quad (2)$$

$$D_{m_f} = \sum_{i=h-2}^{h+2} |I_2(i,v,t) - I_2(i-d_{h_f}, v-d_{v_f}, t+1)| \quad (3)$$

$$D_m = D_{m_b} + D_{m_f} \quad (4)$$

and the weighting factors are given by

$$k_b = \frac{D_{m_f}}{D_{m_b} + D_{m_f}}$$
$$k_f = \frac{D_{m_b}}{D_{m_b} + D_{m_f}}.$$

The final interpolated value obtained using all four methods is obtained as[3]

$$I_3(h,v,t) = k_v I_v(h,v,t) + k_t I_t(h,v,t) + k_o I_o(h,v,t) + k_m I_m(h,v,t)$$

with the weighting factors as

$$k_x = \frac{D_v D_t D_o D_m}{D_x(D_v D_t D_o + D_v D_t D_m + D_v D_o D_m + D_t D_o D_m)}.$$

The final structure of the algorithm is given in Fig. 1.

[3]Note that here k_v, k_t, k_o and k_m correspond to k_{1_3}, k_{2_3}, k_{3_3} and k_{4_3} respectively, while I_m corresponds to $I_{m,4}$.

(a)

(b)

Fig. 3. Comparison of (a) *Us* versus (b) *Hd* for the Megacycles sequence. (a) shows much better rendition of the spatial details while (b) suffers from various artifacts (most notably the displaced lines, which give it an "interlaced" look).

III. Experimental Results

We ran our experiments on two computer-generated sequences, Megacycles and Fish, and one "real-world" sequence, Raft. The first two are interlaced sequences with field size 720×243, while the last one has the field size of 512×240. The sequence source material was progressive and we generated the interlaced versions. These original progressive sequences will be used as a reference point to which we will compare our algorithm. This algorithm arose as we were trying to improve on the one in [6]. We first found that using orientational interpolation as the second level of approximation produced better results than if we just weighted vertical, temporal, and orientational interpolation as in [6]. The final level of approximation, that is, adding motion compensation, did not actually help much, either when we tried the algorithm from [6] or our algorithm. Our hope is that future work might help find better ways of using motion compensation. For these reasons, since we obviously fared better than the algorithm in [6], for comparison purposes we will use the algorithm provided to us by the AT&T HDTV group, which is based on the commercially available deinterlacer by Faroudja and Bialo [9]. This algorithm can be classified as spatiotemporal as well, and will thus be a good reference point.

Fig. 2 shows a frame from each of the three sequences used in the experiments. The Megacycles sequence depicts "robots" moving along checkerboard paths. The Fish sequence depicts a fish on a concentric circle template zooming in, while the Raft sequence shows people on a raft in turbulent waters, and the shore with rocks and trees in the background. The results shown here were obtained using a block size of 8 and a window (search) size of 8 in the motion compensation part. Preliminary experiments show that whenever

the motion estimation is not highly accurate, the algorithm assigns larger weight to the two spatial methods. In particular, orientational interpolation helps in maintaining image resolution, which would be lost if only shift-invariant filters were used. On the other hand, temporal interpolation methods are favored as the vertical velocity component of the motion decreases.

We ran a pilot subjective testing on the three sequences we mentioned above. The test involved 10 subjects (5 expert and 5 nonexpert ones), who were asked to evaluate nine pairs of sequences. For each pair, the subject was asked to make one of the following comparative statements on the scale: "I prefer the right sequence over left one very much," "I prefer the right sequence over left one," "I prefer the right sequence over left one slightly," "I am neutral (cannot decide/no difference)," and the same three statements with right and left reversed. The order in which they saw the sequences was random and was different for each subject. In what follows, we will denote our algorithm by *Us*, reference algorithm by *Hd* and original sequence by *Or*. The subjects saw three pairs (*Us* versus *Hd*, *Us* versus *Or*, *Or* versus *Hd*) for each of the three sequences.

The results of the test are given in Tables I–III. First note that the first column under each sequence name represents the number of people who chose that option, while the second one is the relative score. The relative score is obtained as follows: the choice "prefer very much" gets a 3, "prefer" gets a 2 and "prefer slightly" gets a 1. Neutral does not get a score. Thus, a relative score of 23/30 in Table I means that out of possible 30 (10 people giving "prefer very much" = 3), the algorithm gets 23 (the maximum would be 27 for 9 people who chose this algorithm). In Tables II and III, 0.5 appears since one subject preferred *Us* or *Hd* versus *Or* in one part of the sequence (as the fish is zooming in). The overall conclusion is that in the case of the Megacycles sequence, this algorithm performs very well, and the difference when compared to the reference one is fairly large. As seen in Table II, there were instances when subjects preferred this algorithm even to the original or they saw no difference. One subject preferred the reference algorithm, finding that the slight flicker in *Us* was more annoying than the artifacts in *Hd*. Fig. 3 shows a comparison between this algorithm and the reference one for the Megacycles sequence. For the Fish sequence, the results are still in favor of *Us*, although not that dramatically. The difference between *Us* and *Hd* is still large, and 7 people preferred *Us*. Most of the preference for *Us* comes from sharper edges. However, circular patterns exist in both *Us* and *Hd*. Finally, for the Raft sequence, it seems that both *Us* and *Hd* perform similarly and have similar artifacts. The consensus is that the quality of both is fairly high. Some artifacts can be seen in the raft (people with ores) and in the background (rocks and trees).

REFERENCES

[1] B. Ayazifar and J. S. Lim, "Pel-adaptive model-based interpolation of spatially subsampled images," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, San Francisco, CA, Mar. 1992, pp. III:181–184,

[2] P. Pohjala and M. Karisson, "Line rate upconversion in IDTV applications," *IEEE Trans. Consumer Electron.*, vol. 37, pp. 309–312, Aug. 1991.

[3] D. Gillies, M. Plantholt, and D. Westerkamp, "Motion adaptive field rate upconversion algorithms for 900 lines/100 Hz/2:1 displays," *IEEE Trans. Consumer Electron.*, vol. 36, pp. 149–160, May 1990.

[4] L. Vandendorpe *et al.*, "Motion-compensated conversion from interlaced to progressive formats," *Image Commun. J.*, vol. 6, pp. 193–211, 1994.

[5] E. Dubois and J. Konrad, "Estimation of 2-D motion fields from image sequences with application to motion-compensated processing," in *Motion Analysis and Image Sequence Processing*. Boston, MA: Kluwer, 1993.

[6] A. Nguyen and E. Dubois, "Spatio-temporal adaptive interlaced-to-progressive conversion," *Proc. Int. Workshop on HDTV*, Nov. 1992.

[7] ——, "Spatial directional interpolation using steerable filters," *Proc. Canadian Conf. Electrical and Computer Engineering*, Sept. 1992, pp. MA4.8.1–MA4.8.4.

[8] F. M. Wang, D. Anastassiou, and A. M. Netravali, "Time-recursive deinterlacing for IDTV and pyramid coding," *Image Commun. J.*, vol. 2, pp. 365–374, Oct. 1990.

[9] Y. C. Faroudja and C. A. Bialo, "Method and apparatus for reducing correlated errors in subband coding systems with quantizers," U.S. Patent 5159451, Oct. 27, 1992.

Robust Detection of Skew in Document Images

Avanindra and Subhasis Chaudhuri

***Abstract*—We describe a robust yet fast algorithm for skew detection in binary document images. The method is based on interline cross-correlation in the scanned image. Instead of finding the correlation for the entire image, it is calculated over small regions selected randomly. The proposed method does not require prior segmentation of the document into text and graphics regions. The maximum median of cross-correlation is used as the criterion to obtain the skew, and a Monte Carlo sampling technique is chosen to determine the number of regions over which the correlations have to be calculated. Experimental results on detecting skews in various types of documents containing different linguistic scripts are presented here.**

I. INTRODUCTION

The conversion of paper document to electronic format is routinely performed for records management, archiving, and many other applications. The principal stages in a document conversion system are scanning, binarization, region segmentation, text recognition, and document analysis which includes steps from simple spelling correction to natural language understanding. During the scanning process,the document may not be fed properly into the scanner. The text lines in the image would not be horizontal and may cause problems in segmenting the image to extract its layout structure. For example, in the most commonly used method of profiling [1], in the presence of skew there would not be valleys in the projection histogram and segmentation would not be possible. Skew detection and removal is, thus, a very important stage in document analysis.

Several methods have been developed by researchers for skew angle detection. It can be determined from document edges [2] or text margins. Dengel [1] uses the method of left margin search. The position of the first black pixel in each row of the image is stored in a vector. Using this vector, straight line segments that describe left margin of layout objects (words, text lines, blocks) are determined. The gradient of the straight line defines its angle with respect to the vertical orientation. But this does not provide an

Manuscript received May 25, 1995; revised March 22, 1996. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Maria Petrou.

The authors are with the Department of Electrical Engineering, Indian Institute of Technology, Bombay 400 076, India (e-mail: sc@ee.iitb.ernet.in).

Publisher Item Identifier S 1057-7149(97)00996-2.