Bitmap Resolution Synthesis

Jennifer Guani Dy and Jan P. Allebach

Electronic Imaging Systems Laboratory
Purdue University, School of Electrical and Computer Engineering
West Lafayette, IN 47907-1285
{dy, allebach}@ecn.purdue.edu

1 ABSTRACT

This paper presents an algorithm for interpolating binary images. More specifically it deals with a factor of two interpolation in the horizontal and vertical direction. The algorithm uses a look-up table built on a training set of high and low resolution images. The conditional probabilities of the high resolution pixels based on a corresponding neighborhood of low resolution pixels are used to predict the interpolated image. Then a nonlinear smoothing filter is applied to enhance the resulting image. Since the algorithm is look-up table based, it facilitates simple implementation suitable for printers.

2 INTRODUCTION

As printer resolutions continue to increase, the time to send the bitmap from the host computer to the printer becomes excessive. The usual approach to this memory and transmission problem is to utilize data compression techniques [1, 2, 3]. But these put a high computational burden on the printer. Another way to deal with this problem is to interpolate the low resolution binary image sent from the host to a higher resolution binary image at the printer level.

This paper presents a look-up-table based interpolator which utilizes conditional probabilities obtained by training on a library of low resolution and high resolution images. This approach is reminescent of JBIG's model and adaptive templates block. However, in JBIG the conditional probability tables are built on the sequence of data that have already been sent. The high resolution pixels are predicted only to send the error sequence for compression in coding [1, 4]. In essence, all the low and high resolution image data is availabe to the decoder. In our case, we only have the low resolution binary image. We have to guess what the higher resolution image is. Other predictors based on conditional probabilities used in coding are similar to JBIG in this sense [2, 3, 5].

In Section 3, basic terms used throughout the paper are defined. Then in Section 4 we describe the interpolation algorithm and its different variants. The results and some discussion of the outcomes are also incorporated in this section, as the different versions are introduced. In Section 5, a nonlinear smoothing algorithm for post-processing the interpolated image is presented. Finally, in Section 6 conclusions and future directions are discussed.

3 DEFINITIONS

The following terms will frequently be used:

Model template is a geometric pattern describing the location of pixels relative to the pixels to be estimated.

Context is an integer corresponding to the binary states of the pixels within this template. It is used as an index to the look-up table which provides the values of the estimates.

To measure the performance of the algorithms, the

pixel error rate is obtained. This is the error between the interpolated image and the original high resolution image. The *transition error rate* is also computed to get an idea of what percent of those pixels in error are transition pixels. A "transition" pixel is any pixel for which one or more of its 8 nearest neighbors has a different state than it does. The transition error rate actually gives a better picture of the interpolation error than the pixel error rate.

4 INTERPOLATION ALGORITHMS

4.1 4×4 NONCAUSAL WINDOW

A 4×4 noncausal window as shown in Fig. 1 is used as a template to estimate the four high resolution pixels, a, b, c, and d. In building up the table, the concept of conditional probabilities is utilized. The table is built by training on a set of low resolution and high resolution images. These images are obtained by directly converting postscript files to bit maps at two resolutions, 300 dpi and 600 dpi, using ghostscript. For each low resolution and high resolution image pair, the conditional probability of a given the 4×4 low resolution template is computed. The same is done for b, c, and d. A table of conditional probabilities for every possible context is obtained. The maximum likelihood estimate is used in this computation [6]. Let x_0 , x_1, ..., x_N be a sequence of N independent, identically distributed random variables taking on the values 0 and 1 with probabilities p and $1 - p$. Let k denote the number of zeroes in the sequence. Then, the maximum likelihood estimate of p is $p = k/N$.

The result of this algorithm trained on the text image in Figs. 3 and 4 and then applied to Fig. 3 is shown in Fig. 5. These figures show only a portion of the full image which was an $8\frac{1}{2} \times 11$ in^2 page of 12 pt. text. The pixel error rate is 0.037437 and the transition error rate is 0.298871. The transition error rate is a little high; but visually this error is not too noticeable. A closer inspection of the images reveals that the errors correspond to a one bit displacement of character edges.

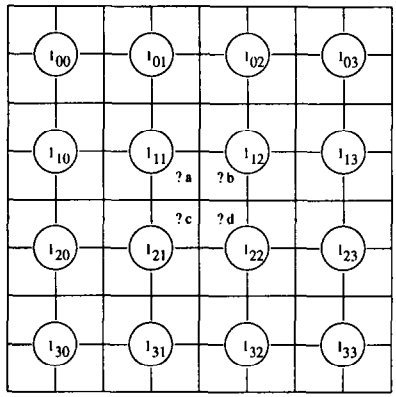

Figure 1: 4×4 template.

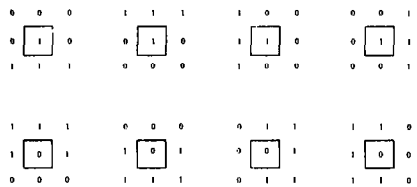

Figure 2: Patterns used for smoothing.

Figure 3: Original 300 dpi image.

Figure 4: Original 600 dpi image.

Figure 5: Interpolated image using 4×4 template.

Figure 6: Smoothed version of interpolated image.

To preliminarily check the effectiveness of this algorithm for cross-training, three different images were used, a times-font image, a script-font and a graphics image. The pixel error rates and transition error rates when trained alone and cross-trained are summarized in Table 1. The transition error rates are relatively high. But visually, the images are quite good except for the graphics image which shows some toning problems.

	TEST IMAGES					
	TIMES		SCRIPT		GRAPHICS	
TRAIN IMG	PER	TER	PER	TER	PER	TER
TIMES	0.014	0.256	0.013	0.275	0.052	0.215
SCRIPT	0.015	0.302	0.011	0.239	0.052	0.215
GRAPHICS	0.018	0.328	0.016	0.312	0.047	0.205
ALL 3	0.014	0.289	0.011	0.242	0.047	0.206

Table 1: Effect of cross-training on prediction error.

4.2 3×3, 5×5, 8×8 NONCAUSAL WINDOWS

Logically we would expect that basing the estimate on more information, i.e. increasing the size of the template, would improve our prediction. The results shown in Table 2 indicate that this is indeed true.

	pixel error rate	trans. error rate
3×3	0.039015	0.305171
4×4	0.037437	0.298871
5×5	0.023648	0.193996
8×8	0.017342	0.141933

Table 2: Effect of template size on prediction error.

These error rates are also reflected visually. However, increasing the size of the template induces other problems. Using a larger window requires a larger training set to insure good performance. Also, a larger table is needed— a table of size 2^{64} for the 8×8 template is needed compared to 2^{16} for a 4×4 template. 2^{64} is approximately 1.84×10^{19} which is impractical for this application. However, the number of actual patterns that occur in typical images is very much less than 2^{64}. A large savings in memory will be obtained by implementing this algorithm as a tree structure [7]. Even for the 4×4 window, this implementation saves memory. For example, for the text image, the number of template patterns that occur is 1245 which is very much less than $2^{16} = 65536$. Each node of the tree is determined by the template. If a 1 is encountered, move to the right node; if 0, move to the left node. The tree has 16 levels for a 4×4 template. At the bottom of the tree are the leafnodes. They contain the conditional probability information. This saves memory because the template patterns not encountered are not stored.

For further memory-space reduction, the tree can be pruned. Pruning means that if the lower levels have no effect on the decision given the higher levels, those lower level nodes are eliminated.

Using the full table for an 8×8 template, 1.84×10^{19} bytes are needed. When a tree structure was used, non-occuring contexts were not stored and only 1.29×10^{8} bytes were used. With pruning, more memory was saved. The tree used 2.3×10^{7} bytes.

5 SMOOTHING ALGORITHM

Taking a closer look at the interpolated text image shown in Fig. 5, we can see that the image is a little bit jagged. This is understandable if we compare

this to the original low resolution image on which this prediction is based. A smoothing algorithm can be applied to the interpolated image to improve its quality. Because the jaggedness is mostly due to one pixel glitches, a nonlinear pattern approach is used. If the 3×3 patterns shown in Fig. 2 occur, the center pixel is modified. The resulting image shown in Fig. 6 significantly improved.

6 CONCLUSION

The algorithms presented are quite simple; however the visual quality of the results is surprisingly good. The algorithms performed much better than pixel replication as can be seen by comparing Figs. 3 and 6. Visually, the quality is almost as good as the original high resolution data printed at 600 dpi. However, there is still room for improvement. The poor performance with graphics images suggests that segmentation prior to interpolation would be desirable. Also, the interpolation algorithm has some problems faithfully producing the curves in the original high resolution image. Extraction of higher level features from the template pixels might help in this respect. Our future research will proceed along these lines.

References

[1] H. Hampel, R. B. Arps, et. al., "Technical Features of the JBIG Standard for Progressive Bi-level Image Compression," *Signal Processing: Image Communication*, Vol. 4, No. 2, April 1992.

[2] T. S. Huang, "Coding of Two-Tone Images," *IEEE Transactions on Communications*, Vol. COM-25, No. 11, November 1977.

[3] A. H. Stadlin, "Coding Techniques for Dithered Binary Images," *Master's Thesis, Electrical Engineering, University of Delaware*, June, 1978.

[4] *CCITT Draft Recommendation T.82 ISO/IEC Draft International Standard 11544 Coded Representation of Picture and Audio Information - Progres sive Bi-level Image Compression*, WG9-S1R5.1, April 3, 1992.

[5] D. L. Duttweiler and C. Chamzas, "Probability Estimation in Arithmetic and Adaptive-Huffman Entropy Coders," *IEEE Transactions on Image Processing*, Vol. 4, No. 3, March 1995.

[6] G. Casella and R. L. Berger, *Statistical Inference*, Pacific Grove, Calif. : Brooks/Cole Pub. Co., c1990.

[7] A. Drozdek, *Data structures in C*, Boston : PWS Pub. Co. , c1995.