Correspondence

Fast Fractal Compression of Greyscale Images

Jean Cardinal

***Abstract*—A new algorithm for fractal compression of greyscale images is presented. It uses some previous results allowing the compression process to be reduced to a nearest neighbors problem, and is essentially based on a geometrical partition of the image block feature space. Experimental comparisons with previously published methods show a significant improvement in speed with no quality loss.**

***Index Terms*—Fractal compression, image coding, tree searching.**

I. Introduction and Previous Work

We begin with a brief summary of the fractal compression method.

Let (M, d) denote a metric space of digital images, where d is a given metric. Let μ_{orig} be the image we want to encode. We try to find a mapping $\tau : M \rightarrow M$, satisfying the following contractivity property:

$$\exists 0 < s < 1, \qquad \forall \mu, \nu \in M,\ d(\tau(\mu), \tau(\nu)) \leq s.d(\mu, \nu) \tag{1}$$

and

$$d(\mu_{orig}, \tau(\mu_{orig})) \simeq 0. \tag{2}$$

This distance is called the Collage error. When the operator τ is applied recursively on any initial picture μ_0, the limit of the sequence $\mu_{i+1} = \tau(\mu_i)$ does exist and is close to μ_{orig}. So μ_{orig} should be close to the fixed point of τ.

In the partitioned iterated functions system (PIFS) [1] theory, the mapping consists of a collection of local blockwise transformations, and each of these transformations is contractive in the image space.

In the original—and still most used—scheme presented in [1], the image is partitioned in so-called *range blocks*, which are nonoverlapping image blocks. We construct a *domain pool*, which is a collection of larger overlapping image blocks. For each range r, we try to find a domain d and a transformation f so that $f(d) \simeq r$. The transformation f consists of two parts: a spatial part, usually a translation and a scaling; and a massic part, modifying the pixels in the block. The massic part is itself usually composed of a rotation or a symmetry, and of a luminance transform, modifying the pixels intensities. Traditionally, the distortion measure used is the mean square error (MSE). In the scheme described in [1], the luminance transform is a first-degree polynomial, so only two coefficients, called respectively the scaling (order-1), and the offset (order-0), have to be computed. To ensure the contractivity of the transformation, the scaling coefficient should have a magnitude less than one.

The problem in the fractal method is that it needs a huge amount of processing time for compression: for each range, the whole domain pool has to be scanned in order to find the best matching domain. Many attempts to reduce the compression time have been made. The first of them was block classification: the search is restricted to domain blocks in the class of the range. For example in [1] blocks are classified using edge informations, and in [2] the quadrants variances of the block are sorted to obtain 72 block classes. But one important step has been made in [3], where the best match search process is reduced to a nearest neighbor search in a suitable metric space. In this technique, each block is associated with a feature vector so that minimizing the collage error between the range and the domain is equivalent to minimizing the distance between the corresponding feature vectors. This is possible if the feature vector is chosen invariant to block transformations. The search for a best match then simply consists of finding the nearest vector in the vector space. Note that this result is exact only when we ignore the contractivity constraint: practically, we will have to find more than one vector close enough to the query. This can however be done much faster than the linear search using—as suggested in [3]—a k–d tree structure (first described in [4]).

Similar ideas are exploited in, e.g., [5] and [6]. In a recent book [7], another related method is described, close to the one we describe in this paper, although less sophisticated. These techniques clearly outperform the classical methods relying on a reduction of the size of the domain pool, either by classification or using some heuristic, like local searching. For more insights on the efficiency of these different classes of algorithms, one can refer to [7]. Finally, such techniques may be compared to those used in vector quantization, like the one presented in [8], which is in turn closely related to [9].

We present here a new algorithm allowing even more acceleration in the compression process. Although it is closely related to more classical tree structures, it appears to be faster than the previous methods.

II. General Description

As described above, the problem of seeking the best transformations can be reduced to the classical computational geometry problem of finding nearest neighbors in a Euclidean space.[1]

Our algorithm takes two sets of image blocks as parameters: the domain blocks set and the range blocks set. All the feature vectors—or keys—corresponding to each block in the two sets are computed: the feature vectors may be, for instance, given by the Saupe's projection operator ϕ. This operator was called the *normalized projection operator*, and was first introduced in [2, Appendix]. It is written

$$\phi(x) = \frac{x - \langle x, e \rangle . e}{\| x - \langle x, e \rangle . e \|} \tag{3}$$

where $e = (1, \cdots, 1)/\sqrt{k}$, k is the number of dimensions, and $\langle ., . \rangle$ denotes the dot product. We assume that image blocks have the same size (i.e., the domain blocks have been downsized), and we treat them as simple vectors in $\Re^k$. Actually, applying this operator is equivalent to making x both zero-sum and normalized. In [6], a similar operator is defined in the frequency domain, and in [3], the operator is generalized to the case where fixed basis blocks are used instead of a simple offset term. Our approach, described in the following, is applicable to those cases as well.

The next phase consists in recursively partitioning the search space, containing both the range keys and the domain keys, until a sufficiently

Manuscript received August 18, 1998; revised August 9, 2000. This work was supported by a grant from the National Scientific Research Fund (FNRS). The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Rashid Ansari.

The author is with the Computer Science Department, Brussels Free University, B-1050 Brussels, Belgium (e-mail: jcardin@ulb.ac.be).

Publisher Item Identifier S 1057-7149(01)00096-3.

[1]The search space should be defined as Euclidean due to scalar products used in the following developments.

small number u of domain keys is left in each subspace. Each of the ranges corresponding to the keys left in a subspace is then compared to each of the domains corresponding to the remaining domain keys, and the best transformation for each range is saved. Sometimes the process can end up with a subspace containing no domain at all. This exception case must be treated separately, for example by canceling the previous partition. The information used for the partitioning need not be saved, which means that the technique does not use much memory space.

The core of the algorithm is, of course, contained in the partitioning procedure. One possible way is given by the so-called optimal k–d tree construction algorithm (see [4]): the space is partitioned along a hyperplane dividing the key set in two halves, each containing the same number of keys, and orthogonal to the axis for which the distribution of the keys projections has the widest spread. This solution, also used in [7], is easy to handle, especially when we have to backtrack through the levels, but since we only perform a descent in the tree we can use a more efficient technique.

A generalization of this method consists of finding a hyperplane with no restriction on its direction.

In the method presented, the search space is recursively divided in two parts by a $(k-1)$-dimensional hyperplane. At each step, the dividing hyperplane may be represented by a normalized vector $v \in \Re^k$ and a scalar value $\mu \in \Re$, where the hyperplane is orthogonal to v and contains the point $\mu.v$. The problem is then to find the hyperplane by finding the corresponding v and μ at each step. Each key will be on one side of the hyperplane or the other, depending on whether $\langle p,\ v\rangle$ is larger or smaller than μ. An illustration may be found in Fig. 1.

The complexity of the partitioning process is $O(k(N_R + N_D))$ for each level, where N_R and N_D are the number of ranges and domains, respectively.

A good solution for finding v has been proposed in [9], where the direction of the plane is given by the principal eigenvector v_{opt} of the covariance matrix of the keys distribution. We will define this solution as optimal, because this direction maximizes the residual variance of the keys, i.e., it is the most discriminant one-dimensional axis we can find.

In our algorithm we introduce a heuristic in order to find a good cutting hyperplane without having to compute v_{opt}, which may be time-consuming, especially in high dimensions. The method is formally described as follows.

- Let $\{r_i \in \Re^k \mid i = 1 \cdots N_R\}$ and $\{d_i \in \Re^k \mid i = 1 \cdots N_D\}$ be the feature vectors sets for the ranges and the domains.
- Compute the gravity center of the range distribution: $g = (1/N_R)\sum_{i=1}^{N_R} r_i$.
- For each $i \in 1 \cdots N_R$, compute the projection of r_i on the unit sphere centered on g: $s_i = (r_i - g)/\parallel r_i - g \parallel$.
- The partitioning hyperplane will be orthogonal to the direction given by $v = w/\parallel w\parallel$, where: $w = (1/N_R)\sum_{i=1}^{N_R} s_i$, and will contain the point g.

The value corresponding to μ is given by $\langle g,\ v\rangle$.

The goal of the heuristic is to find a significant direction in the cloud of points. We first take the gravity center of the points in order to center the distribution, and then simply take the average of the directions "observed" from this point. This very simple technique is reminiscent of the iteration proposed in [10] to compute a good approximation of the principal component. Note that if $\|w\| = 0$, we must choose an arbitrary vector. This case, however, has never been encountered during our experiments.

When the optimal direction v_{opt} is used, this method also tends to maximize the average distance between each range key and the border of the cell containing it, and minimizes the probability of having an interesting neighbor in an adjacent cell. We expect our heuristic to have the same property.

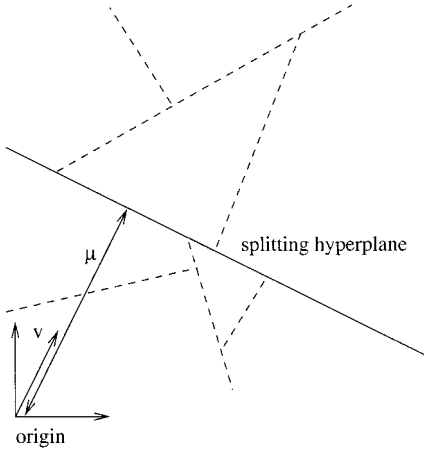


Fig. 1. Illustration of the partitioning process: dashed lines represent the subsequent splitting hyperplanes. A domain-range pair is tested only if the corresponding keys fall in the same cell.

In our method, we assume that the range keys spatial distribution is similar to the domain keys one. This assumption can be considered as a corollary of the local self-similarity property exploited in the PIFS model. The algorithm takes this idea into account, and only uses the ranges to find the partitioning hyperplanes: ranges usually being less numerous than domains, the calculation is faster.

III. Experimental Comparison

Briefly, three methods will be tested, each one corresponding to a different choice for v:

1) random method: v is chosen randomly;
2) optimal method: $v = v_{opt}$;
3) heuristic method: v is chosen from the heuristic described above.

Note that in the optimal case, the calculation of v_{opt} is also restricted to the ranges keys. In each case, μ is the average of the distribution of the projections of the keys on v: the calculation is faster than for the median, and it gives similar results.

The three methods above are compared to the classical k–d tree algorithm.

The compression scheme used in the tests is quite simple: the image partition is a three-level quadtree with range blocks sizes of fours, eight, and 16 pixels. The domain pool is constituted by domains twice as big as ranges, and aligned on a 2×2 pixels lattice. No square isometry (block rotations or symmetries) is used in the transformations.

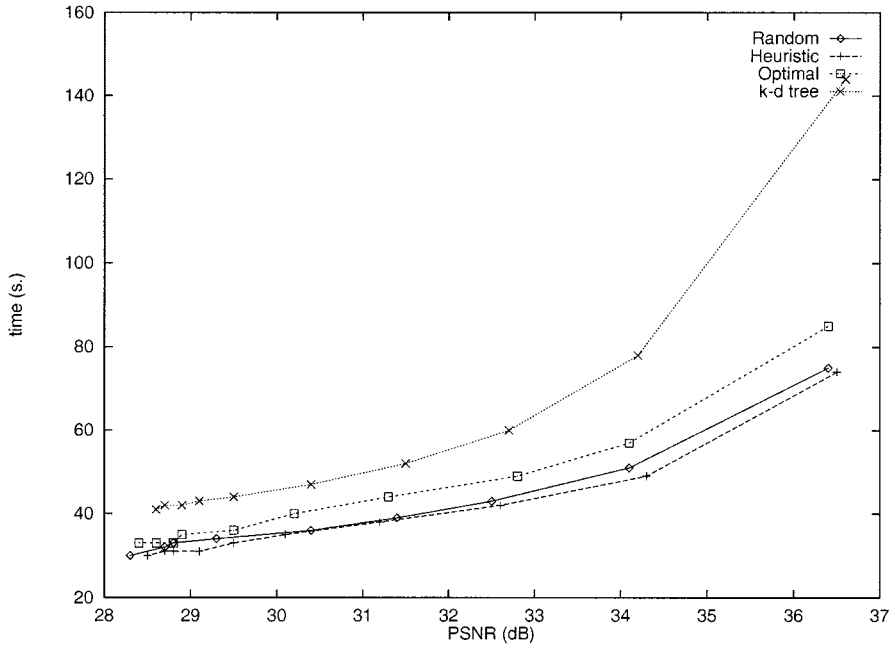
The keys are computed using the ϕ operator.[2]

A simple uniform quantization has been used to store the coefficients of the tranformations: seven bits were used for the offset (order-0 coefficient), and five for the scaling (order-1 coefficient). The domain pool index is not quantized, so that the number of bits used by the index is the same for all the transformations in one level of the tree, and equal to $\lceil \log(\text{domain pool size}) \rceil$.

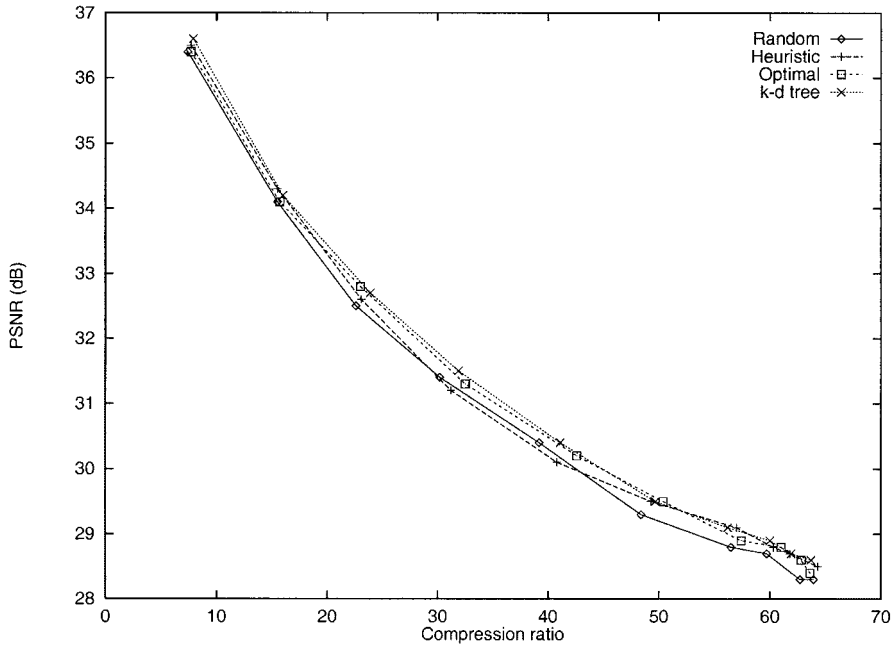
The computation of the eigenvector in the optimal method is done using a fast implementation of the Jacobi diagonalization algorithm.

The k–d tree search algorithm has been optimized too: instead of searching for the nearest neighbor, we look for a $(1+\epsilon)$-nearest neighbor. (This is a point whose distance to the query is less than $(1+\epsilon)$ times the distance of the nearest neighbor.) It is implemented as a simple relaxation of the backtracking condition called the *ball-overlap-bounds test* in [4]. We have set $\epsilon = 3$: This optimization

[2]Normally, for each block x, two keys must be inserted: $\phi(x)$ and $-\phi(x)$. For our tests, only the first key is used: this has the same effect as reducing the domain pool by half.



(a)



(b)

Fig. 2. Results for Zelda: (a) timings and (b) quality.

leads to a great improvement in speed and only a small precision loss. In our new algorithms, we have also found $u = 300$ to be optimal for the problem (u is the maximum number of direct domain comparisons).

The algorithms have been tested on 12 images of the standard greyset available at the Waterloo Bragzone website[3] using a Pentium-based machine. The size of the images is 512×512, with eight bits per pixel.

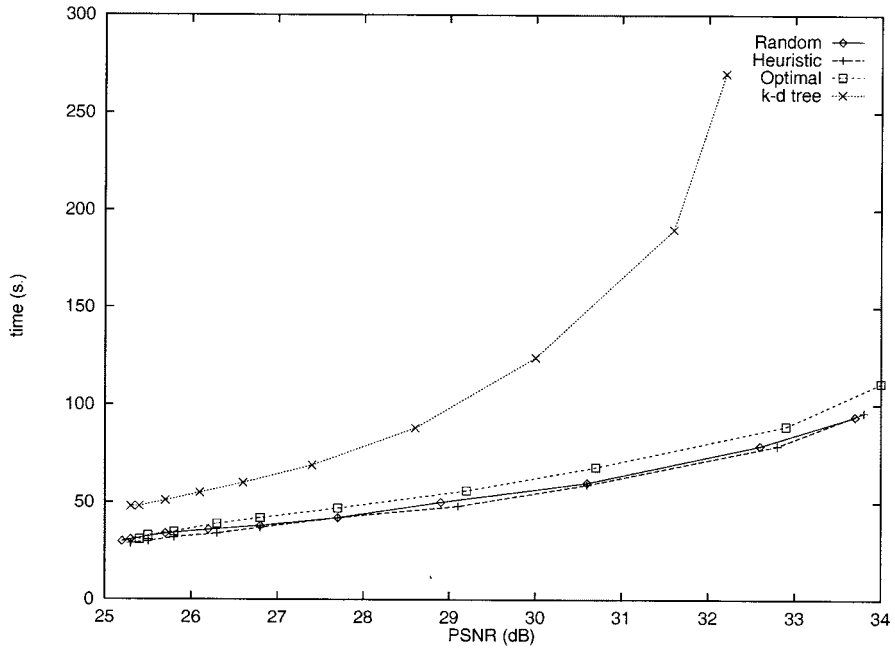
IV. Results

Graphs of compression ratios and compression times are given for two images: Zelda and Goldhill (Figs. 2 and 3). The quality is measured using the peak-signal-to-noise ratio. Two general graphs are also showed, for which the values have been averaged on the basis of the whole image set (Fig. 4).

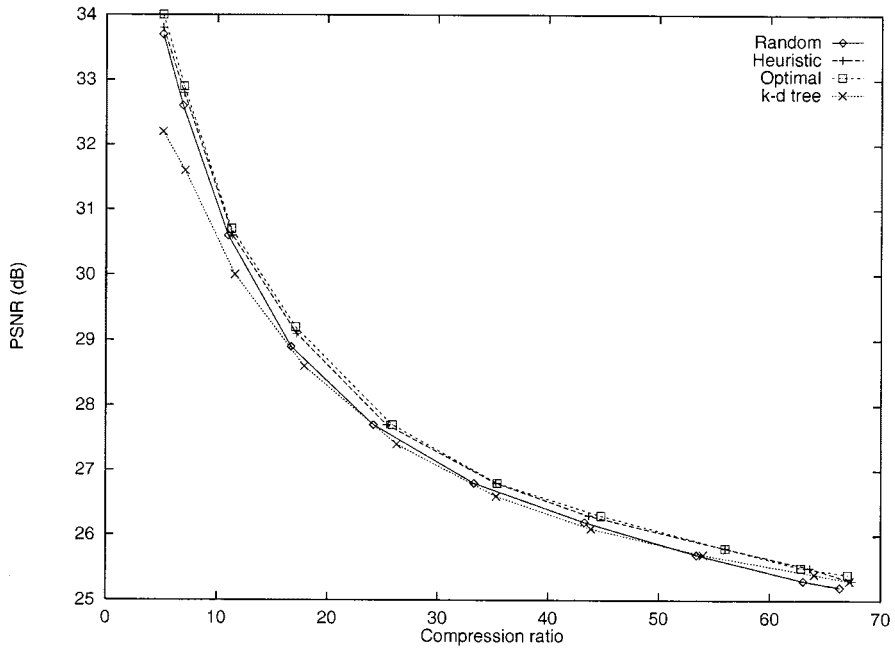
We can say that

- three new methods are always faster than the k–d tree method. Acceleration ratio is varying between 1.5 and 3;
- optimal method gives better quality results compared to the random method;
- time loss using the optimal method compared to the random and heuristic methods is around 20%;
- on the average, for a fixed encoding time, the k-d tree method gives no better quality results than the three other ones, and even worse results for low compression ratios;

[3]http://links.uwaterloo.ca



(a)



(b)

Fig. 3. Results for Goldhill: (a) timings and (b) quality.

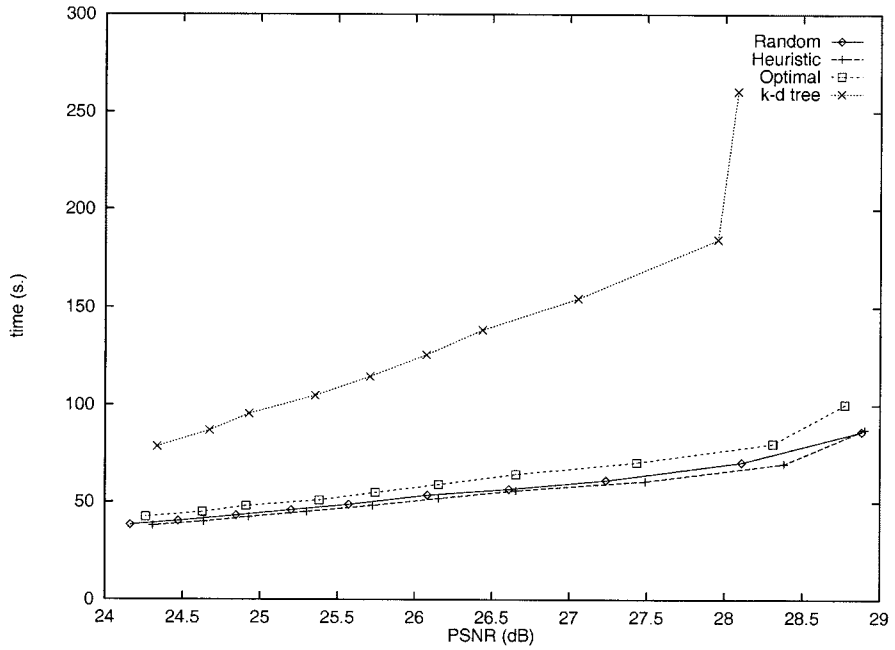
- use of the heuristic induces no time loss compared to the random method and gives better results, close to the optimal method ones.

Note that the speed of the algorithms is measured versus the quality: the random method, for instance, actually takes more time for a fixed degradation.

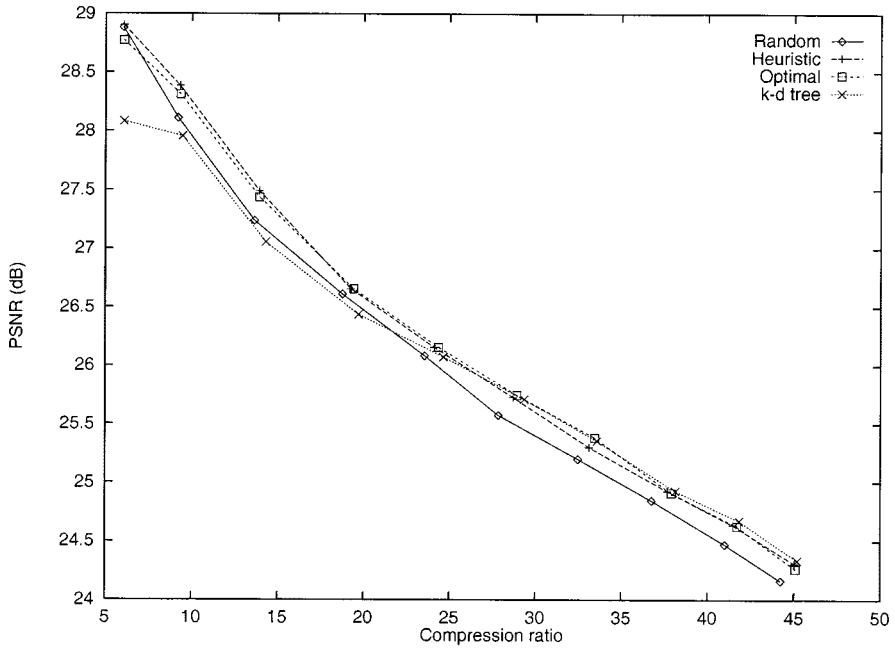
V. Conclusion

Our new algorithm for fractal codes searching is simple to implement, faster than the classical methods and space-economical. The speed of the algorithm is essentially due to the fact that there is no backtracking in the hierarchical partitioning structure. This backtracking process introduces a $O(2^k)$ constant in the complexity of the k-d tree search. We successfully compensate the loss of precision by a more meaningful partitioning of the search space. The fact that the partition is adapted to the range blocks distribution only also contributes to the efficiency of the search. This method may also be seen as a classification scheme where each bucket represents a class.

Further experiments and refinements can be tested: for example we have not taken into account the fact that the feature vectors $\phi(x)$ all belong to the a unit $(k-1)$-dimensional hypersphere. We could also have applied a dimensionality reduction technique such as in [6], or an overlapping cells principle, like in [7].



(a)



(b)

Fig. 4. Averaged results: (a) timings and (b) quality.

ACKNOWLEDGMENT

The author would like to thank G. Lauria, Y. Roggeman, D. Saupe, B. Wohlberg, and the referees for their helpful comments and ideas about this text.

REFERENCES

[1] A. Jacquin, "A fractal theory of iterated Markov operators with applications to digital image coding," Ph.D. dissertation, Georgia Inst. Technol., Atlanta, Aug. 1989.

[2] Y. Fisher, *Fractal Image Compression—Theory and Application*. New York: Springer-Verlag, 1994.

[3] D. Saupe, "Accelerating fractal image compression by multi-dimensional nearest-neighbor search," in *Proc. DCC'95*, Mar. 1995.

[4] J. L. Bentley, R. A. Finkel, and J. H. Friedman, "An algorithm for finding best matches in logarithmic expected time," *ACM Trans. Math. Softw.*, vol. 3, no. 3, pp. 209–226, 1977.

[5] J. Kominek, "Algorithm for fast fractal image compression," in *Proc. IS&T/SPIE 1995 Symp. Electronic Imaging: Science Technology*, vol. 2419, 1995.

[6] B. E. Wohlberg and G. de Jager, "Fast image domain fractal compression by DCT domain block matching," *Electron. Lett.*, vol. 31, pp. 869–870, 1995.

[7] N. Lu, *Fractal Imaging*. New York: Academic, 1997.

[8] I. Katsavounidis, C.-C. Jay Kuo, and Z. Zhang, "Fast tree-structured nearest neighbor encoding for vector quantization," *IEEE Trans. Image Processing*, vol. 5, pp. 398–404, Feb. 1996.

[9] R. F. Sproull, "Refinements to nearest-neighbor searching in k-dimensional trees," *Algorithmica*, vol. 6, pp. 579–589, 1991.
[10] M. Partridge and R. A. Calvo, "Fast dimensionality reduction and simple PCA," *Intell. Data Anal.*, vol. 2, Aug. 1998.

Low-Bit-Rate Video Coding Using Dense Motion Field and Uncovered Background Prediction

K. P. Lim, M. N. Chong, and A. Das

***Abstract*—It is well known that accurate dense motion field can improve the video coding efficiency. This paper presents a novel Markov random field (MRF) model that estimates both the dense motion and uncovered background fields in image sequences, and the application of these estimates in H.263-based video coding framework.**

***Index Terms*—Dense motion field, H.263, Markov random field, video coding.**

I. Introduction

It has been found that coding efficiency is improved by applying dense motion field (DMF) to video coding algorithm [1], [2], [11]. In [1] and [11], modified a Horn–Schunck [3] algorithm is used to compute DMF, whereas in [2], Markov random field (MRF) is adopted. These approaches [1], [2], [11] suffer a major drawback for relying heavily on motion information for image reconstruction. Therefore, when new objects appear in the video scene, they are unable to encode the new information efficiently [10].

In this paper, we present a novel MRF model to improve the dense motion field estimation by overcoming the problem of occlusion. In addition, a new video-coding framework that uses dense motion field and the property of motion continuum for video coding without incurring huge motion overheads is presented.

II. Dense Motion Field Estimation

Occlusion is an inherent problem for DMF estimation since motion is not defined in the occluded regions. The proposed MRF model addresses this problem by using a novel approach to estimate both the occlusion and dense motion fields based on the duality principle of occlusion; the occluded region is the uncovered region if the frame sequence is viewed in the reversed direction and it is true vice-versa.

A. MAP Formulation

Let $\boldsymbol{g_k}$ denote the kth frame. Let $\boldsymbol{d}_{k-1,\,k}(\vec{x})$ be the motion vector on the spatial location $\vec{x}$ in $\boldsymbol{g_{k-1}}$ indicating a corresponding point on $\vec{x} + \boldsymbol{d}_{k-1,\,k}(\vec{x})$ in $\boldsymbol{g_k}$ lying on the same motion trajectory. To model occlusion field accurately using the duality of occlusion, the MRFs are modeled in a bi-directional Bayesian framework that is different from the work reported in [4]–[6]. The new MRF models for the motion and occlusion fields are derived as follows.

Let

$\boldsymbol{d_f}=\boldsymbol{d_{k-1,\,k}}$	forward motion field;
$\boldsymbol{u_f}=\boldsymbol{u_{k-1,\,k}}$	forward uncovered background label field where $\boldsymbol{u_f}(\vec{x}) = 1$ indicates that site $\vec{x}$ contains an uncovered background pixel in frame $\boldsymbol{g_{k-1}}$ and $\boldsymbol{u_f}(\vec{x}) = 0$ indicates otherwise;
$\boldsymbol{d_b}=\boldsymbol{d_{k,\,k-1}}$	backward motion field;
$\boldsymbol{u_b}=\boldsymbol{u_{k,\,k-1}}$	backward uncovered background label field where $\boldsymbol{u_b}(\vec{x}) = 1$ indicates that site $\vec{x}$ contains an uncovered background pixel in frame $\boldsymbol{g_k}$ and $\boldsymbol{u_b}(\vec{x}) = 0$ indicates otherwise.

Hence, given the observed frames $\boldsymbol{g_{k-1}}$ and $\boldsymbol{g_k}$, the MAP estimates can be written as

$$(\hat{\boldsymbol{u}}_f,\, \hat{\boldsymbol{d}}_f,\, \hat{\boldsymbol{u}}_b,\, \hat{\boldsymbol{d}}_b) = \arg \max_{u_f,\, d_f,\, u_b,\, d_b} p(\boldsymbol{u}_f,\, \boldsymbol{d}_f,\, \boldsymbol{u}_b,\, \boldsymbol{d}_b|\boldsymbol{g}_k,\, \boldsymbol{g}_{k-1}). \tag{1}$$

Direct relaxation of the four unknown fields is too complex to be solved; (1) is estimated by solving the forward and backward fields separately. Using Bayes rule, (1) can be formulated in two ways. The first is to condition the backward fields $\boldsymbol{u_b}$ and $\boldsymbol{d_b}$ and, frame $\boldsymbol{g_{k-1}}$ in (1). This leads to the MAP estimates of the forward fields

$$\begin{aligned}(\hat{\boldsymbol{u}}_f,\, \hat{\boldsymbol{d}}_f) = \arg \max_{u_f,\, d_f} \{&p(\boldsymbol{g}_k|\boldsymbol{u}_f,\, \boldsymbol{d}_f,\, \boldsymbol{u}_b,\, \boldsymbol{d}_b,\, \boldsymbol{g}_{k-1}) \\ &\cdot p(\boldsymbol{u}_f|\boldsymbol{d}_f,\, \boldsymbol{u}_b,\, \boldsymbol{d}_b,\, \boldsymbol{g}_{k-1}) \\ &\cdot p(\boldsymbol{d}_f|\boldsymbol{u}_b,\, \boldsymbol{d}_b,\, \boldsymbol{g}_{k-1}) p(\boldsymbol{u}_b,\, \boldsymbol{d}_b|\boldsymbol{g}_{k-1})\}.\end{aligned} \tag{2}$$

The MAP estimates of the backward fields are derived by conditioning (1) with the forward fields, $\boldsymbol{u_f}$, $\boldsymbol{d_f}$, and $\boldsymbol{g_k}$. This leads to the other MAP estimates of the backward fields

$$\begin{aligned}(\hat{\boldsymbol{u}}_b,\, \hat{\boldsymbol{d}}_b) = \arg \max_{u_b,\, d_b} \{&p(\boldsymbol{g}_{k-1}|\boldsymbol{u}_b,\, \boldsymbol{d}_b,\, \boldsymbol{u}_f,\, \boldsymbol{d}_f,\, \boldsymbol{g}_k) \\ &\cdot p(\boldsymbol{u}_b|\boldsymbol{d}_b,\, \boldsymbol{u}_f,\, \boldsymbol{d}_f,\, \boldsymbol{g}_k) \\ &\cdot p(\boldsymbol{d}_b|\boldsymbol{u}_f,\, \boldsymbol{d}_f,\, \boldsymbol{g}_k) p(\boldsymbol{u}_f,\, \boldsymbol{d}_f|\boldsymbol{g}_k)\}.\end{aligned} \tag{3}$$

By solving (2) and (3) alternately, the MAP of (1) can be estimated. From (2), the likelihood is modeled as a Gaussian distribution with zero mean and variance σ^2. Let $\lambda = \mathbf{1}/2\sigma^2$, N be the number of sites and φ the set of all sites in the image, the potential function [9] of the likelihood is given as

$$\begin{aligned}&U_u(\boldsymbol{g}_k|\boldsymbol{u}_f,\, \boldsymbol{d}_f,\, \boldsymbol{u}_b,\, \boldsymbol{d}_b,\, \boldsymbol{g}_{k-1}) \\ &\quad = \frac{1}{2N} \ln(2\pi\sigma^2) + \lambda \sum_{\vec{x} \in \varphi} [\boldsymbol{g}_k(\vec{x} + \boldsymbol{d}_f(\vec{x})) - \boldsymbol{g}_{k-1}(\vec{x})]^2.\end{aligned} \tag{4}$$

Using the Hammersley–Clifford Theorem [7], Gibbs distributions are used to specify the MRFs. The motion field *prior* model consists of singleton and doubleton clique potential functions

$$V_d^c(\boldsymbol{d}_f|\boldsymbol{u}_b,\, \boldsymbol{d}_b,\, \boldsymbol{g}_{k-1}) = V_d^{c1}(\boldsymbol{d}_f) + V_d^{c2}(\boldsymbol{d}_f|\boldsymbol{u}_b). \tag{5}$$

The singleton clique potential function is defined as follows:

$$V_d^{c1}(\boldsymbol{d}_f(\vec{x}_i)) = \lambda_0(1 - 2\delta[\boldsymbol{d}_f(\vec{x}_i)]\boldsymbol{c}[\vec{x}_i]) \tag{6}$$

Manuscript received December 21, 1998; revised June 20, 2000. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Rashid Ansari.

The authors are with the School of Applied Science, Nanyang Technological University, Singapore 639798 (e-mail: asmnchong@ntu.edu.sg).

Publisher Item Identifier S 1057-7149(01)01179-4.