SIAM J. SCI. COMPUT.
Vol. 27, No. 5, pp. 1573–1593

ADAPTIVE MULTISCALE REDISTRIBUTION FOR VECTOR QUANTIZATION*

YAIR KOREN† AND IRAD YAVNEH†

Abstract. Vector quantization is a classical problem that appears in many fields. Unfortunately, the quantization problem is generally nonconvex, and therefore affords many local minima. The main problem is finding an initial approximation which is close to a "good" local minimum. Once such an approximation is found, the Lloyd–Max method may be used to reach a local minimum near it. In recent years, much improvement has been made with respect to reducing the computational costs of quantization algorithms, whereas the task of finding better initial approximations received somewhat less attention. We present a novel multiscale iterative scheme for the quantization problem. The scheme is based on redistributing the representation levels among aggregates of decision regions at changing scales. The rule governing the redistribution relies on the so-called point density function and on the number of representation levels in each aggregate. Our method focuses on achieving better local minima than those achieved by other contemporary methods such as LBG. When quantizing signals with sparse and patchy histograms, as may occur in color images, for example, the improvement in distortion relative to LBG may be arbitrarily large.

Key words. quantization, Lloyd–Max, LBG, point density

AMS subject classifications. 62H30, 65H10, 65K10, 65U05, 68U10

DOI. 10.1137/040607769

1. Introduction.

Quantization [11, 13] is the process of representing continuum with only a finite number of representatives or representing an initially rich amount of discrete data with a lesser amount of representatives. Rounding off real numbers to the nearest integer is a simple form of scalar quantization. Representing color images with a lesser amount of colors is an example of vector quantization.

In general, many digital signal processing problems require some form of quantization. In particular, as the world moves toward an increased use of digital media, the digital processing of sound, images, and video involves an increasing amount of quantization. Some problems that would benefit from better quantization algorithms are analogue-to-digital conversion, sampling in general, and compression [16, 15].

Quantization and Voronoi tessellations are closely related and have a variety of applications to more than just signal processing and telecommunication. For example, distribution of resources and clustering algorithms [14, 19, 18] are, in essence, quantization algorithms that are used in various fields such as statistical analysis, pattern recognition, learning theory, computer graphics, and combinatorial chemistry [5]. A review on Voronoi tessellations as well as additional applications, such as quadrature rules, finite difference schemes, statistics, cellular biology, and the cellular behavior of animals [17], can be found in [7].

1.1. Formal definition.

Stating the problem more formally, let Ω be some input domain and let $p : \Omega \to \mathbb{R}$ be the probability density function of some random process over Ω. Let n be a known positive integer. A quantizer $q : \Omega \to \Omega$ is defined

*Received by the editors May 5, 2004; accepted for publication (in revised form) December 21, 2004; published electronically February 3, 2006. This research was supported by the Israeli Science Foundation under grant 48/02 and by the Israeli Ministry of Science.
http://www.siam.org/journals/sisc/27-5/60776.html
†Department of Computer Science, Technion—Israel Institute of Technology, Haifa 32000, Israel (yair_k@cs.technion.ac.il, irad@cs.technion.ac.il).

by n representation levels $R = \{\vec{r}_i\}_{i=1}^n$ and n decision regions $\{D_i\}_{i=1}^n$, where for all $i, r_i \in D_i$, and $\cup_i D_i = \Omega$. That is, $q(\Omega)$ is a piecewise constant approximation of Ω, where all $\vec{x} \in D_i$ are represented by $\vec{r}_i$. An optimal quantizer achieves minimal distortion, defined as the expectation of the quantization error raised to some power t,

$$\mathcal{D}(q) = E[\|\vec{x} - q(\vec{x})\|_2^t] = \sum_{i=1}^{n} \int_{D_i} \|\vec{x} - \vec{r}_i\|_2^t \, p(\vec{x}) \, d\vec{x}. \tag{1.1}$$

Here, $\|\cdot\|_2$ is the L_2 norm. In the discrete case, where $p(\vec{x})$ is nonzero at only a finite number of points $\{\vec{x}_i\}_{i=1}^N$, a similar functional is minimized:

$$\mathcal{D}(q) = E[\|\vec{x} - q(\vec{x})\|_2^t] = \sum_{i=1}^{n} \left\{ \sum_{\vec{x}_j \in D_i} \|\vec{x}_j - \vec{r}_i\|_2^t p(\vec{x}_j) \right\}. \tag{1.2}$$

Henceforth, we permit $p(\vec{x})$ to be any nonnegative function, not necessarily a distribution density function of unit mass. We also set $t = 2$, yielding the ubiquitous mean square error (MSE) measure. In the theoretical background section we show that our algorithms are equally applicable for any t.

1.2. Previous work. In recent years, many methods have been presented that reduce the complexity of finding a solution, e.g., solving multiple 1-D problems [1], recursively splitting the domain in two [25, 16, 30], and accelerating known algorithms [32, 27]. These methods are usually greedy and/or heuristic and involve a complexity/distortion tradeoff. That is, lower complexity algorithms generally arrive at solutions with higher distortion. Wu [31] generalizes the algorithms described in [25, 16, 30] and attempts to arrive at a good local minimum by making an observation about a constant number of consecutive splits and by using dynamic programming.

1.3. The Lloyd–Max iterative process. Necessary conditions for a minimum of (1.1) are

$$\begin{aligned} \frac{\partial}{\partial \vec{r}_i} \mathcal{D}(q) &= 0, \qquad 1 \le i \le n, \\ \frac{\partial}{\partial D_i} \mathcal{D}(q) &= 0, \qquad 1 \le i \le n. \end{aligned} \tag{1.3}$$

When $t = 2$, these conditions yield

$$\begin{aligned} \vec{r}_i &= \frac{\int_{D_i} \vec{x} p(\vec{x}) d\vec{x}}{\int_{D_i} p(\vec{x}) d\vec{x}}, \qquad 1 \le i \le n, \\ D_i &= \mathcal{V}(\vec{r}_i), \qquad\qquad 1 \le i \le n. \end{aligned} \tag{1.4}$$

Here, $\mathcal{V}(\vec{r}_i)$ is the Voronoi cell corresponding to $\vec{r}_i$, i.e., the set of all points in Ω that are closer to $\vec{r}_i$ than to any other representation level $\vec{r}_j$, $j \neq i$. Lloyd [22] and Max [24] independently developed (1.4) and proposed "ping-ponging" back and forth

between the two optimality equations,

$$D_i^\ell = \mathcal{V}((\vec{r}_i)^\ell), \qquad \ell = 0, 1, 2, \ldots, \tag{1.5}$$

$$(\vec{r}_i)^{\ell+1} = \frac{\int_{D_i^\ell} \vec{x} p(\vec{x}) d\vec{x}}{\int_{D_i^\ell} p(\vec{x}) d\vec{x}}, \quad \ell = 0, 1, 2, \ldots, \tag{1.6}$$

that is, optimizing the D_i's according to the $\vec{r}_i$'s (computing the Voronoi cells of the $\vec{r}_i$'s) followed by optimizing the $\vec{r}_i$'s according to the new D_i's (computing the centers of mass of the D_i's), and so on. $\vec{r}^0$ is an initial guess at the solution. This process converges monotonically to a solution that satisfies (1.4) (see [22]).

When applying Lloyd–Max iterations (1.5), (1.6) in the discrete case (1.2) one need not actually compute the Voronoi diagram, which takes $\theta(n \log n + n^{\lceil k/2 \rceil})$ operations for a k-dimensional domain Ω. For high dimensions it is both sufficient and efficient to determine, for each $\vec{x} \in \Omega$, which representation level $\vec{r}_i$ is closest to $\vec{x}$. The naive method of inspecting all n possibilities for each $\vec{x} \in \Omega$ and choosing the closest one takes $O(kNn)$ operations, where $N = |\{\vec{x} : p(\vec{x}) \neq 0\}|$. If a k-d tree [9] is used to find best matches, then the expected computation time of a Lloyd–Max iteration is $O(kN \log n)$.

For scalar quantization ($k = 1$), Fleischer showed that a sufficient condition for a unique minimum of the distortion is that $\log(p(x))$ be concave [8]. For vector quantization, we are unaware of a similar result.

Since the Lloyd–Max iterative process requires an initial guess at the solution in order to start iterating, many heuristic methods, such as k-means [23] and Linde–Buzo–Gray (LBG) [21], have been proposed to find a good initial approximation. The LBG algorithm [21] is a "grid refinement" type of process that starts by finding a solution (using the generalized Lloyd algorithm) for two $\vec{r}_i$'s and then splitting each $\vec{r}_i$ into two new $\vec{r}_i$'s. This process is repeated recursively until the number of $\vec{r}_i$'s equals n. The generalized Lloyd algorithm is the iterative process (1.5), (1.6) with the reduction in distortion used as a stopping criterion. That is, the process is stopped at the iteration in which the ratio between the current distortion and the previous iteration's distortion exceeds some threshold. In what follows, we shall compare our method with LBG.

A multigrid [4, 29] approach for scalar (1-D) quantization was presented in [20]. Our approach was basically a multigrid full approximation scheme algorithm [3] with the Lloyd–Max iteration (1.5), (1.6) used as relaxation. The benefits are three-fold. First, much faster convergence is achieved. The convergence rates of our method seem independent of the number of representation levels sought. In contrast, both Lloyd–Max and LBG converge at a rate that is quadratically dependent on the number of representation levels. Second, the speed up alleviates the need for a convergence/stopping criterion such as the one used for LBG. This is because reaching machine accuracy requires only several iterations when using our method. Third, our method sometimes arrives at better minima than other methods.

The vector quantization problem is generally nonconvex and therefore affords multiple local minima [12]. Figure 1.1 depicts several local minima for the simple case, where $p(x, y) \equiv 1$ and four representation levels are sought. We consider the acceleration of convergence less important than converging to a solution of lower distortion. This paper presents methods for both scalar and vector quantization to overcome the multiple local minima problem. Our methods arrive at solutions of lower distortion and do so in less time compared to competing algorithms.

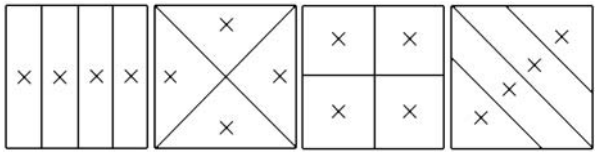

FIG. 1.1. *Multiple local minima for $p(x, y) \equiv 1$. These are all local minima, as the Lloyd–Max criteria* (1.4) *are satisfied. That is, the representation levels are the centers of mass of the decision regions and the decision regions are the Voronoi diagram of the representation levels.*

This paper is organized as follows. Section 2 presents the theoretical background upon which our algorithms are built. Section 3 provides a detailed description of our proposed adaptive multiscale redistribution (AMR) algorithms for vector quantization. Section 4 includes results of our vector quantization algorithms and a comparison with LBG. Section 5 presents a conclusion and further discussion.

2. Theoretical background. Much theoretical research and analysis have been conducted on the problem of quantization [2, 22, 33, 26, 10, 28]. In this section, in order to maintain a self-contained presentation, we recount Gersho's theoretical results concerning optimality criteria for vector quantizers [10]. The optimality criteria are valid for general values of t in the distortion definition (1.1). Therefore, the algorithms derived in the following section are also valid for any t. Gersho's results are true in the large n limit. Nevertheless, our algorithms, which are based on Gersho's results, perform extremely well for small n as well. Examples are presented in section 4.

In his dissertation [33], Zador proves that the *local* structure of an asymptotically optimal quantizer can be that of an optimal quantizer for a uniform distribution. Therefore, we henceforth assume that $p(\vec{x})$ is uniform, whereas in what follows we shall apply the optimality criteria derived in this section to nonuniform $p(\vec{x})$ as well.

In 1979, Gersho [10] made the now widely accepted conjecture that, for a uniform distribution $p(\vec{x})$ in any dimension k, the optimal decision regions are all congruent to some k-dimensional convex polytope (precluding the decision regions touching the boundaries of Ω). The Voronoi diagram in this case is referred to as a *lattice tessellation* of the k-dimensional domain Ω. Gersho's conjecture has been proven for one (see [6, p. 59]) and two [28] dimensions, where the appropriate tessellation-generating polytopes are the section and the regular hexagon, respectively.

Henceforth, we assume that Gersho's conjecture is correct. Under this assumption, an analysis of the distortion of lattice quantizers is within reach. The class of admissible polytopes in $\mathbb{R}^k$, denoted H_k, is defined as follows. A convex polytope H is admissible, i.e., in H_k, if it generates a tessellation of $\mathbb{R}^k$ that is a Voronoi diagram with respect to the centers of mass of the polytopes comprising this tessellation.

The *normalized inertia* of a polytope H is defined as

$$I(H) = \int_H \|\vec{x} - \mathtt{com}\,(H)\|^t \, d\vec{x} \Big/ V\,(H)^{1+t/k}, \tag{2.1}$$

where $V(H)$ is the volume of H and $\mathtt{com}(H)$ is the center of mass of H, or more formally, the point y in H that minimizes $\int_H \|x - y\|^t dx$. Note that $I(\alpha H) = I(H)$ for all $\alpha > 0$, where $\alpha H = \{\alpha\vec{x} : \vec{x} \in H\}$. Intuitively, the normalized inertia measures the quality of a polytope's shape, disregarding size, with respect to distortion minimization.

An optimal polytope, H^*, is an admissible polytope that minimizes the normalized inertia, i.e.,

$$I(H^*) = \inf_{H \in H_k} I(H). \tag{2.2}$$

Given a quantizer q_n having n representation levels, a piecewise constant *point density function* [22] for q_n is defined as

$$\lambda_n(\vec{x}) = (nV(D_i))^{-1} = \frac{1}{n \int_{D_i} d\vec{y}}, \tag{2.3}$$

where D_i is the decision region containing $\vec{x}$, and $V(D_i)$ is D_i's volume. $\lambda_n(\vec{x})\Delta V(\vec{x})$ approximates the fraction of representation levels in the volume element $\Delta V(\vec{x})$. As n approaches infinity, $\lambda_n(\vec{x})$ approximates closely a continuous density function, denoted $\lambda(\vec{x})$. $\lambda(\vec{x})$ has unit mass, i.e., $\int_\Omega \lambda(\vec{x})d\vec{x} = 1$, since $\int_{D_i} \lambda_n(\vec{x})d\vec{x} = 1/n$ for any i and n. Integrating $\lambda(\vec{x})$ over a set $S \in \Omega$ yields the fraction of representation levels in S, whereas integrating $\lambda_n(\vec{x})$ over S approximates this fraction. That is,

$$\frac{|\{\vec{r}_i \,:\, \vec{r}_i \in S\}|}{n} \approx \int_S \lambda_n(\vec{x})d\vec{x}. \tag{2.4}$$

When the number of representation levels is large we may assume that $p(\vec{x}) \approx p(\vec{r}_i)$ for all $\vec{x} \in D_i$; that is, we may assume that the distribution is approximately uniform inside each decision region. Under this assumption, the probability may be removed from under the integral in the distortion definition (1.1), yielding

$$\mathcal{D}(q_n) = \sum_{i=1}^{n} p(\vec{r}_i) \int_{D_i} \|\vec{x} - \vec{r}_i\|^t \, d\vec{x}. \tag{2.5}$$

Plugging in the normalized inertia, we obtain

$$\mathcal{D}(q_n) = \sum_{i=1}^{n} p(\vec{r}_i)\, I(H^*)\, [V(D_i)]^{1+\frac{t}{k}}, \tag{2.6}$$

since, under Gersho's conjecture, the minimal distortion is achieved by decision regions that are similar to H^*. According to (2.3), $V(D_i)$ may be expressed by $\lambda_n(\vec{x})$:

$$\mathcal{D}(q_n) = n^{-(t/k)} I(H^*) \sum_{i=1}^{n} p(\vec{r}_i) \left[\lambda_n(\vec{r}_i)\right]^{-(t/k)} V(D_i). \tag{2.7}$$

As n approaches infinity, the above sum is approximated by the integral

$$\mathcal{D}(q_n) \approx n^{-(t/k)} I(H^*) \int_\Omega \frac{p(\vec{x})}{[\lambda(\vec{x})]^{t/k}} d\vec{x}. \tag{2.8}$$

Equation (2.8) is an approximation of the distortion that is independent of the representation levels and the decision regions. In fact, the only unknown in (2.8) is $\lambda(\vec{x})$. Therefore, the minimal distortion is achieved by optimally choosing $\lambda(\vec{x})$, i.e., by minimizing the integral.

Observe that, by Hölder's inequality,

$$\int_\Omega \left\{ \left[p\left(\vec{x}\right) \lambda(\vec{x})^{-(t/k)} \right]^{\frac{1}{1+t/k}} \right\} \left\{ \lambda(\vec{x})^{\frac{t/k}{1+t/k}} \right\} d\vec{x} \tag{2.9}$$

$$\leq \left\{ \int_\Omega p\left(\vec{x}\right) \lambda(\vec{x})^{-(t/k)} d\vec{x} \right\}^{\frac{1}{1+t/k}} \left\{ \int_\Omega \lambda\left(\vec{x}\right) d\vec{x} \right\}^{\frac{t/k}{1+t/k}},$$

with equality if and only if the right-hand side integrands are proportional, i.e., if and only if $p(\vec{x})[\lambda(\vec{x})]^{-(t/k)} \propto \lambda(\vec{x})$.

As $\lambda(\vec{x})$ has unit mass, we obtain from (2.9)

$$\int_\Omega \frac{p(\vec{x})}{\left[\lambda\left(\vec{x}\right)\right]^{t/k}} d\vec{x} \geq \left\{ \int_\Omega \left[p\left(\vec{x}\right)\right]^{\frac{1}{1+t/k}} d\vec{x} \right\}^{1+t/k}. \tag{2.10}$$

The left-hand side of (2.10) is evidently minimized when equal to the right-hand side. By (2.8), (2.9), and (2.10) the distortion is therefore minimized if and only if

$$\lambda(x) \propto [p(x)]^{k/(k+t)}. \tag{2.11}$$

In what follows we construct algorithms based on the two following corollaries, appearing in Gersho's paper [10].

COROLLARY 2.1. *An optimal quantizer's point density function obeys*

$$\int_S \lambda(\vec{x}) d\vec{x} \approx \frac{\int_S p^{\frac{k}{k+t}}\left(\vec{x}\right) d\vec{x}}{\int_\Omega p^{\frac{k}{k+t}}\left(\vec{x}\right) d\vec{x}} \tag{2.12}$$

for any measurable set S in Ω.

Proof. This is a direct result of the previous derivations. The right-hand side denominator assures unit volume for $\lambda(\vec{x})$. ∎

COROLLARY 2.2. *For an optimal quantizer q, each decision region makes an equal contribution to the distortion* (1.1).

Proof. Using (2.3) and (2.11), one may express both $p(\vec{r}_i)$ and $V(D_i)$ by $\lambda(\vec{x})$ in (2.7). This yields the stated result, as each term in the sum in (2.7) is reduced to the same constant, namely, $1/n$ times the proportion constant from (2.11) ∎

The property indicated by Corollary 2.2 was observed for $k = 1$ by Panter and Dite [26] and for $k = 2$ by Toth [28].

3. Adaptive multiscale redistribution algorithms. We now exploit the theoretical results described above within a novel AMR algorithm. We begin by defining three methods for building a quantizer. Our ultimate quantization algorithm involves concatenating these three methods.

The first method is an extremely quick way of attaining a good initial set of n representation levels. This method heavily relies on (2.12) and achieves a distortion that usually is nearly as low as our final result. The complexity of obtaining this initial guess is proportional to $\log(n)$ Lloyd–Max relaxations, where n is the number of representation levels sought. Stopping this iterative initial guess phase after two to three iterations usually results in a good initial approximation, which may be even better than LBG's final result. From our tests, LBG [21] also seems to require $O(\log(n))$ iterations in order to arrive at an initial guess, but LBG's guess is usually not as good as the one provided by our method, as shall be demonstrated in section 4. In

addition, stopping LBG before $O(\log(n))$ Lloyd–Max iterations have been performed usually results in a poor initial approximation.

The second method is used to improve an initial guess. The improvement is based on Corollary 2.2 and is performed by moving representation levels from low distortion areas to areas with higher distortion.

The last method is simply applying Lloyd–Max iterations (1.5), (1.6) to the best approximation found to date, thus reaching a local minimum of the quantization distortion. We propose a quantization algorithm that is made up of these three steps. First, find an initial guess. Second, improve the initial guess. Third, relax until a local minimum has been reached.

3.1. The general approach. In both acquiring an initial guess and improving it, we use the following AMR approach. A hierarchy of scales is defined. Scale m, denoted R^m, is a partition of the set of representation levels R into m disjoint aggregates. That is,

$$R^m = \{R_j^m\}_{j=1}^m, \; R = \bigcup_j R_j^m. \tag{3.1}$$

At any scale m, all m aggregates are approximately equal in size. At the finest scale, each aggregate is made up of a single representation level, that is, $m = n$. When passing from scale m to scale $m/2$ (the next coarser scale), the m aggregates are grouped in pairs to form approximately $m/2$ new aggregates. This set of $m/2$ new aggregates is scale $m/2$. The coarsest scale consists of two aggregates.

At each scale, we determine the poorest aggregate, that is, the aggregate that would benefit the most from an additional representation level. In a similar manner, we determine the richest aggregate, that is, the aggregate that would suffer the least from giving up a representation level. Then we transfer a representation level from the richest aggregate to the poorest aggregate.

3.2. Acquiring an initial guess. For the purpose of acquiring an initial guess we exploit our knowledge of the optimal point density function (2.12). Let

$$D(R_j^m) = \bigcup_{\vec{r}_i \in R_j^m} D_i, \tag{3.2}$$

that is, $D(R_j^m)$ is the union of the decision regions of all the representation levels in R_j^m. For each aggregate R_j^m of the m aggregates in R^m, (2.12) suggests the fractional number of representation levels that an optimal quantizer allocates to the region $D(R_j^m)$. Subtracting this amount (multiplied by n) from the actual number of representation levels in $D(R_j^m)$, we arrive at the *wealth*, $\mathcal{W}_j^m$, of aggregate R_j^m. More formally,

$$\mathcal{W}_j^m = |R_j^m| - n \left\{ \frac{\int_{D(R_j^m)} p^{\frac{k}{k+2}}(\vec{x}) d\vec{x}}{\int_\Omega p^{\frac{k}{k+2}}(\vec{x})\, d\vec{x}} \right\}. \tag{3.3}$$

According to this definition, the wealth of an aggregate of decision regions is simply the excess of representation levels it currently contains. We call aggregates whose wealth is above zero *rich* aggregates and aggregates whose wealth is below zero *poor* aggregates.

Our algorithm for obtaining an initial guess is as follows. We initialize R^n by randomly choosing n representation levels uniformly in Ω. We use this uniform initial approximation for two reasons. First, it is extremely cheap to compute. Second, and more important, the $Obtain-Guess$ algorithm transfers representation levels to places in Ω that already contain other representation levels. Therefore, a uniform initial approximation gives an equal chance for all areas in Ω to receive representation levels. Next, we call the following recursive function with R^n as input (note that $n = |R^n|$).

ALGORITHM.
$R^m \leftarrow Obtain - Guess(p(\vec{x}), R^m)$.

1. Perform a single Lloyd–Max relaxation.
2. Calculate $\mathcal{W}^m = \{\mathcal{W}_i^m\}_{i=1}^m$.
3. Let $\mathcal{W}_{max}^m$ denote $\max(\mathcal{W}^m)$ and $\mathcal{W}_{min}^m$ denote $\min(\mathcal{W}^m)$.
4. *WHILE* $\mathcal{W}_{max}^m - \mathcal{W}_{min}^m > 1$, *DO*:
   (a) $\vec{r}_j \leftarrow Select - Source(R_{max}^m)$.
   (b) $\vec{t} \leftarrow Select - Destination(R_{min}^m)$.
   (c) Transfer $\vec{r}_j$ from R_{max}^m to R_{min}^m at location $\vec{t}$.
   (d) $\mathcal{W}_{min}^m \leftarrow \mathcal{W}_{min}^m + 1$, $\mathcal{W}_{max}^m \leftarrow \mathcal{W}_{max}^m - 1$.
5. $R^{m/2} \leftarrow Match(R^m)$.
6. *IF* $R^{m/2} = R^m$
   (a) *RETURN* R^m.
7. *ELSE*,
   (a) Recursively call $Obtain - Guess(p(\vec{x}), R^{m/2})$.

Throughout the $Obtain - Guess$ algorithm, subscripts max and min refer to aggregates of maximal and minimal wealth, respectively. Note that the decision regions are not recalculated after the transition of representation levels, but only within the Lloyd–Max relaxation occurring at step 1 of $Obtain - Guess$. This single Lloyd–Max relaxation at each scale is performed to improve the solution. In addition, this relaxation is needed for the calculation of $\mathcal{W}_i^m$, as this calculation requires knowing the decision regions. The output of the $Obtain - Guess$ algorithm is the set of representation levels yielding the lowest distortion achieved.

Note that $Obtain-Guess$ is an adaptive multiscale redistribution of the representation levels. The goal of this redistribution is to have the quantizer's point density function obey (2.12) as best as possible. Since checking (2.12) for all possible sets S is intractable, we do so only for various unions of decision regions. The following lemma shows how each transition of a representation level at step 4(c) of $Obtain - Guess$ makes the point density function of the quantizer more coherent with (2.12).

LEMMA 3.1. *When* $\mathcal{W}_{max}^m - \mathcal{W}_{min}^m > 1$ *(step* 4 *of* $Obtain - Guess$*), the transition of a representation level (step* 4(c) *of* $Obtain - Guess$*) reduces* $\sum_i |\mathcal{W}_i^m|$.

Proof. We shall prove that $|\mathcal{W}_{max}^m - 1| + |\mathcal{W}_{min}^m + 1| < |\mathcal{W}_{max}^m| + |\mathcal{W}_{min}^m|$. Assume that $0 \le \mathcal{W}_{max}^m < 1$ and $-1 < \mathcal{W}_{min}^m \le 0$. When this is not true, the result is trivial. Note that $\mathcal{W}_{max}^m \ge 0$ and $\mathcal{W}_{min}^m \le 0$ always hold. $|\mathcal{W}_{max}^m - 1| = 1 - \mathcal{W}_{max}^m < -\mathcal{W}_{min}^m = |\mathcal{W}_{min}^m|$. The first equality is implied by $\mathcal{W}_{max}^m < 1$ and the inequality by $\mathcal{W}_{max}^m - \mathcal{W}_{min}^m > 1$. In a similar manner, $|\mathcal{W}_{min}^m + 1| = 1 + \mathcal{W}_{min}^m < \mathcal{W}_{max}^m = |\mathcal{W}_{max}^m|$, where the first equality is implied by $\mathcal{W}_{min}^m > -1$ and the inequality by $\mathcal{W}_{max}^m - \mathcal{W}_{min}^m > 1$. ∎

The selection of a representation level in the richest aggregate, at step 4(a) of $Obtain - Guess$, is described next.

ALGORITHM.
$\vec{r}_i \leftarrow Select - Source(R_j^m)$.

1. *IF* $\left|R_j^m\right| > 1$
   (a) *IF* $R_j^m = R_s^{2m}$ for some $1 \le s \le 2m$,
      i. Recursively call $Select - Source(R_s^{2m})$.
   (b) *ELSE*,
      i. $R_j^m = R_u^{2m} \cup R_v^{2m}$ for some $1 \le u, v \le 2m$.
      Let R_ℓ^{2m} be the aggregate that achieves $\max\{\mathcal{W}_u^{2m}, \mathcal{W}_v^{2m}\}$.
      ii. Recursively call $Select - Source(R_\ell^{2m})$.
2. *ELSE*,
   (a) *RETURN* the single $\vec{r}_i \in R_j^m$.

Less formally, we wish to remove the representation level in R_j^m that belongs to the richest aggregate *at all scales*. We traverse the scales from the present scale to the finer scales. At each scale m we deal with a single aggregate R_j^m. R_j^m is usually made up of two $2m$-scale aggregates (the actual matching is performed by $Match$). We choose the richer aggregate of the two and continue recursively. When we reach an aggregate containing a single representation level, we choose it as the representation level to remove.

The selection of a destination in the poorest aggregate, at step 4(b) of $Obtain - Guess$, is described next.

ALGORITHM.
$\vec{t} \leftarrow Select - Destination(R_j^m)$.

1. *IF* $\left|R_j^m\right| > 2$
   (a) *IF* $R_j^m = R_s^{2m}$ for some $1 \le s \le 2m$,
      i. Recursively call $Select - Destination(R_s^{2m})$.
   (b) *ELSE*,
      i. $R_j^m = R_u^{2m} \cup R_v^{2m}$ for some $1 \le u, v \le 2m$.
      Let R_ℓ^{2m} be the aggregate that achieves $\min\{\mathcal{W}_u^{2m}, \mathcal{W}_v^{2m}\}$.
      ii. Recursively call $Select - Source(R_\ell^{2m})$.
2. *ELSE*,
   (a) *RETURN* the mean of the set R_j^m.

Less formally, we wish to insert a representation level at a location that belongs to the poorest aggregate *at all scales*. In a manner similar to $Select - Source$, we traverse the scales from the present scale to the finer scales. This time, at each scale m we choose the poorest aggregate of the two that make up R_j^m and continue recursively. When we reach an aggregate containing just two representation levels, we choose the destination as the midpoint between the two levels.

The coarser scales are required to take care of the situations in which many small aggregates are moderately poor and many other small aggregates are moderately rich. In this case, as the wealth function (3.3) is linear, the grouping of several somewhat poor aggregates should form a larger aggregate that is poor enough to warrant an additional representation level. In a similar fashion, the grouping of several somewhat rich aggregates should form a larger aggregate that is rich enough to allow a

representation level to be removed. In the appendix, we demonstrate the importance of wrapping the redistribution in a multiscale framework.

In light of the above, the matching between pairs of aggregates that is performed at step 5 of $Obtain-Guess$ and is designed to produce larger aggregates should only match aggregates that are either both rich or both poor. In addition we prefer to match aggregates that are close to each other. This is because adding a new representation level to an aggregate made up of two distant regions only increases the wealth of the region that accepts the representation level. This accepting region may not even be one of the two distant ones, in which case the transition is clearly not the right course of action. If the accepting region is one of the two distant ones and indeed the transfer is beneficial, then it should have been performed at a finer scale, where each of the two regions was a separate aggregate.

We define the distance between two aggregates as the distance between the mean of the representation levels in each aggregate, i.e.,

$$\|R_i^m, R_j^m\| = \left\| \frac{\sum_{\vec{r}_i \in R_i^m} (\vec{r}_i)}{|R_i^m|} - \frac{\sum_{\vec{r}_j \in R_j^m} (\vec{r}_j)}{|R_j^m|} \right\|_2 . \tag{3.4}$$

More formally, the matching algorithm is defined as follows.

ALGORITHM.
$R^{m/2} \leftarrow Match(R^m)$.

1. Initialize $R^{m/2} = \phi$.
2. Sort the aggregates in R^m according to $|\mathcal{W}_i^m|$.
3. Traverse the sorted list, *FOR EACH* aggregate R_j^m, *DO*:
   (a) Denote by Δ the set of unmatched aggregates, R_k^m, that satisfy $\text{sign}(\mathcal{W}_k^m) = \text{sign}(\mathcal{W}_j^m)$.
   (b) *IF* $\Delta = \phi$
      i. Add R_j^m to $R^{m/2}$ and mark R_j^m as matched.
   (c) *ELSE*,
      i. Find $R_i^m = \underset{R_k^m \in \Delta}{\text{argmin}} \|R_k^m, R_j^m\|$.
      ii. Add $R_j^m \cup R_i^m$ to $R^{m/2}$ and mark both R_j^m and R_i^m as matched.
4. *RETURN* $R^{m/2}$.

Using k-d trees [9] to compute $\text{argmin} \|R_k^m, R_j^m\|$ in step 3(c)(i) of $Match$, the expected complexity of the matching algorithm is $O(km \log m)$, where m is the number of aggregates. Note that this is asymptotically much less than the $O(kN \log n)$ required for a Lloyd–Max relaxation (recall that N is the number of points, $\vec{x}$, for which $p(\vec{x}) > 0$).

We apply $Obtain-Guess$ several times until no representation levels are transferred at any scale. Each call to $Obtain-Guess$ applies $O(\log(n))$ Lloyd–Max relaxations that are the bulk of its time complexity. In practice we have applied this method to many examples. We have found that about 2–3 calls to $Obtain-Guess$ generate a "steady" initial guess, i.e., one that is not affected by another call to $Obtain-Guess$.

From our tests, the distortion of the initial guess thus obtained may be quite close to the final distortion achieved by concatenating our three methods. Whether or not the subsequent gain justifies the additional work is both application and user

dependent. For example, video compression may be performed off-line and the additional computational cost may therefore by acceptable in return for an even mild improvement in the resulting compressed file size. On the other hand, real time applications do not have the luxury of long processing time even if the improvement in the resulting distortion is significant. We expect those who need a good solution and require a fast algorithm to be quite satisfied with the initial guess and not apply the following two phases. On the other hand, those who require the best solution possible and/or are willing to accept a somewhat slower process are encouraged to apply the entire three phase process. For $p(\vec{x})$ that are sparse and patchy (e.g., color images), the ratio between the distortion achieved by AMR (either the single phase algorithm or the three phase algorithm) and the distortion achieved by LBG may be arbitrarily small, as shall be demonstrated in section 4.

3.3. Improving an initial guess. Improving a guess requires more leniency than acquiring an initial guess. The harsh demand that $\max(\mathcal{W}^m) - \min(\mathcal{W}^m) > 1$ (step 4 of $Obtain - Guess$) in order to perform a transition is relaxed by $\max(\mathcal{W}^m) - \min(\mathcal{W}^m) > T$ for some positive threshold T. In all of our tests, we have used $T = 1/2$. In addition we redefine an aggregate's wealth. This time the definition is based on Corollary 2.2,

$$\mathcal{W}_j^m = 1 - \frac{average\ distortion\ at\ aggregate\ j}{average\ distortion\ in\ \Omega}, \tag{3.5}$$

where "*average distortion*" is the average distortion per representation level.

ALGORITHM.
$R^m \leftarrow Improve - Guess(R^m, T, \nu)$.

1. Perform a single Lloyd–Max relaxation.
2. *WHILE* the distortion of R^m, $\mathcal{D}(R^m)$, decreases, *DO:*
   (a) Calculate $\mathcal{W}^m = \{\mathcal{W}_i^m\}_{i=1}^m$.
   (b) *IF* $\max(\mathcal{W}^m) - \min(\mathcal{W}^m) > T$,
      i. Move 1 representation level from the richest aggregate to the poorest aggregate, using $Select - Source$ and $Select - Destination$. Denote by $\tilde{R}^m$ the set of aggregates after the transition.
      ii. $\tilde{R}^m \leftarrow Beat - Target - Distortion(\tilde{R}^m, \mathcal{D}(R^m), \nu)$.
      iii. *IF* the distortion has not improved,
         A. discard the new $\tilde{R}^m$ and keep R^m.
      iv. *ELSE*,
         A. $R^m \leftarrow \tilde{R}^m$.
3. Form a set of new aggregates: $R^{m/2} \leftarrow Match(R^m)$.
4. *IF* $R^{m/2} = R^m$
   (a) *RETURN* R^m.
5. *ELSE*,
   (a) Recursively call $Improve - Guess(R^{m/2}, T, \nu)$.

According to Corollary 2.2, an optimal quantizer obeys $\mathcal{W}_j^m = 0$ for all $1 \le j \le m \le n$. This definition still permits us to call aggregates whose wealth is above zero *rich* aggregates and call aggregates whose wealth is below zero *poor* aggregates.

Our improvement method uses the above distortion-based wealth definition (3.5). From our tests, the distortion-based wealth definition (3.5) leads to better results. On the other hand, a meaningful threshold for the initial guess phase (step 4 in

$Obtain-Guess$), such as the one that Lemma 3.1 provides for (3.3), is more difficult to come by for (3.5). This is why we use the point density–based wealth definition (3.3) with threshold $T = 1$ when acquiring an initial guess.

$Beat-Target-Distortion$ applies Lloyd–Max relaxations to $\tilde{R}^m$, attempting to achieve a distortion lower than $\mathcal{D}(R^m)$. After each relaxation, the function tries to assess if ν additional relaxations are likely to reduce the distortion enough. This assessment is made by assuming that each additional Lloyd–Max relaxation will reduce the distortion by the actual reduction achieved via the previous Lloyd–Max relaxation.

The linear extrapolation we use to forecast the distortion reduction has the following property. It is optimistic in the early stages, at which time a more accurate approximation may halt the process prematurely, whereas in the later stages, as the distortion reduction rate reduces, it becomes more accurate. If the target distortion is beaten, then $Beat-Target-Distortion$ applies ν additional Lloyd–Max relaxations and the result is returned. The additional ν relaxations are performed to allow the new solution to realize its potential. More formally, we have the following algorithm.

ALGORITHM.
$R^m \leftarrow Beat-Target-Distortion(R^m, \mathcal{D}, \nu)$.

1. *DO*: {
   (a) Denote the current distortion $\mathcal{D}^{old} = \mathcal{D}(R^m)$.
   (b) Perform a Lloyd–Max relaxation on R^m.
   (c) Denote the new distortion by $\mathcal{D}^{new} = \mathcal{D}(R^m)$.
   (d) *IF* $\mathcal{D}^{new} < \mathcal{D}$
      i. Apply ν additional Lloyd–Max relaxations to R^m and return the result.

   } *WHILE* $\mathcal{D}^{new} - \nu|\mathcal{D}^{new} - \mathcal{D}^{old}| < \mathcal{D}$.
2. *RETURN* R^m.

When a representation level is added or removed (step 2(b)(i) of $Improve-Guess$), a Lloyd–Max iteration (1.5), (1.6) propagates this change only to the neighbors of this representation level. Assuming that the representation levels are spread uniformly in the domain Ω, we see that it would take approximately $n^{1/k}$ Lloyd–Max iterations to propagate this change throughout all of the representation levels. For this reason, we used $\nu = \lceil n^{1/k} \rceil$ throughout our tests. For the threshold T, we used $T = 1/2$.

Coarse scale changes may upset the balance achieved at the finer scales. Therefore, it may be beneficial to call $Improve-Guess$ several times. Before calling $Improve-Guess$ an additional time, ν may be raised and T lowered. In our examples we always called $Improve-Guess$ two to three times, while T and ν were never changed.

The AMR process can be stopped at any time. In our tests, no matter when the process was stopped, the results were better than those achieved by LBG after the same amount of time.

4. Vector quantization results. Many digital signal processing applications involve some form of quantization. As an example, color images may be compressed by representing them with only a fraction of their initial amount of colors. The histograms of color images are usually sparse and patchy, as Figure 4.1 illustrates. Close inspection of Figure 4.1 reveals that the image's histogram is concentrated around seven patches. Six of the patches are fairly close to one another and represent the brighter areas in the image, e.g., sky, lawn, and walls and roof of the house. The seventh patch is quite distant from the rest and represents the darker areas in the

image, e.g., the house's window shutters, the bushes surrounding the house, and the shadows of the car and the utility pole. We demonstrate our algorithms on discrete 2-D data using the discrete distortion definition (1.2) with $t = 2$. Let

$$p_{ij} = \begin{cases} 5 & i, j \in \{1, 1, 2, \ldots, 150\}, \\ 3 & i, j \in \{201, 201, 202, \ldots, 256\}, \\ 0 & otherwise. \end{cases} \tag{4.1}$$

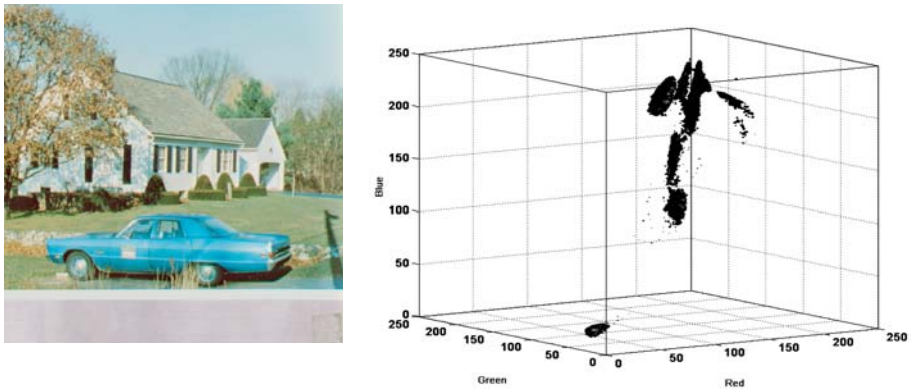

FIG. 4.1. *Left: House and car,* 512×512 *color image. Right: An isosurface of the image's histogram* $p(\vec{x})$.

p_{ij} is a simple probability density function comprised of only two patches (see Figure 4.2). Nevertheless, this p_{ij} is quite similar to probability density functions of real color images, such as the one depicted in Figure 4.1. Therefore, we consider the results exhibited in this section for the above 2-D naive p_{ij} as evidence of our methods' abilities to quantize real multidimensional signals.

From p_{ij}'s structure alone we can deduce that most of the representation levels should be allocated to the larger patch. The fact that the patches are square is completely unimportant; we have defined p_{ij} in this way solely for simplicity.

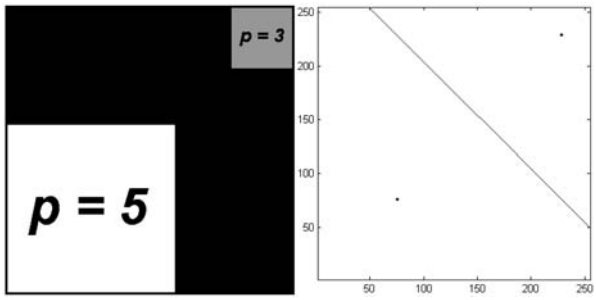

FIG. 4.2. *Left:* p_{ij}. *Right: Globally optimal solution for* $n = 2$ *produced by both LBG and AMR.*

We applied both LBG and AMR to this problem. Each algorithm execution in this section is performed independently of the others, that is, each execution starts from scratch. LBG was used with threshold $1 - 10^{-2}$, that is, Lloyd–Max iterations were

performed at each scale until the ratio of successive distortions ($\mathcal{D}^{new}/\mathcal{D}^{old}$) exceeded $1-10^{-2}$. On the finest scale, we have taken as LBG's initial guess the solution at the iteration in which the ratio of successive distortions exceeded this $1-10^{-2}$ threshold. Thereafter, Lloyd–Max relaxations were performed until the solution reached machine accuracy, that is, until no further improvement could be made. At the stages in which the representation levels are split, LBG adds a new representation level next to each existing one. Our implementation of LBG inserts the new representation levels in a random direction and at a random short distance relative to each of the existing representation levels. An implementation in which the new representation levels are inserted at a constant direction and distance from the existing ones would permit the building of simple probability density functions, for which LBG performs quite poorly.

For $n = 2$ both LBG and our method produce the same solution, namely, one representation level in each patch (see Figure 4.2). Figure 4.3 displays the solutions achieved by AMR and LBG for $n = 4, 8, 16, 32$, and 64. Figure 4.3 implies that LBG cannot always remedy errors made at coarse scales and that coarse scale solutions are not always good initial approximations for the finer scales, an assumption implicitly made by LBG. At the stage when $n = 2$, LBG decided to allocate one representation level to each patch. From that moment on, no representation levels could move from patch to patch since, for any representation level, the center of mass of its decision region is always located within its patch.

TABLE 4.1

LBG and AMR results when applied to the p_{ij} that is defined by (4.1) *and depicted in* 4.2. *From left to right, n = number of representation levels.* ***AMR iter.*** *are the cumulative numbers of Lloyd–Max iterations required to complete the first, second, and third phases of the AMR algorithm, respectively.* ***AMR dist.*** *is the distortion of the initial guess (phase one) and of the final result (phase three).* ***LBG iter.*** *are the numbers of Lloyd–Max iterations required to acquire an initial guess and are the numbers of Lloyd–Max iterations applied throughout the LBG algorithm.* ***LBG dist.*** *is the distortion of the initial guess and of the final result achieved by LBG.* ***LBG:AMR*** *are the ratios of the distortions achieved by LBG to those achieved by AMR (both at the initial guess and final result stages). These ratios may be considered as the factor by which AMR is better than LBG.*

n	**AMR iter.**			**AMR dist.**		**LBG iter.**		**LBG dist.**		**LBG:AMR**	
2	5	11	18	$4.73 \cdot 10^8$	$4.27 \cdot 10^8$	16	17	$4.3 \cdot 10^8$	$4.27 \cdot 10^8$	0.91	1.00
4	8	28	60	$1.73 \cdot 10^8$	$1.72 \cdot 10^8$	21	31	$2.72 \cdot 10^8$	$2.67 \cdot 10^8$	1.57	1.56
8	10	38	85	$6.99 \cdot 10^7$	$6.90 \cdot 10^7$	27	29	$1.07 \cdot 10^8$	$1.07 \cdot 10^8$	1.53	1.56
16	12	41	64	$3.41 \cdot 10^7$	$3.32 \cdot 10^7$	37	79	$5.73 \cdot 10^7$	$5.47 \cdot 10^7$	1.68	1.65
32	14	60	130	$1.68 \cdot 10^7$	$1.63 \cdot 10^7$	43	83	$2.87 \cdot 10^7$	$2.66 \cdot 10^7$	1.71	1.64
64	17	88	190	$8.20 \cdot 10^6$	$8.07 \cdot 10^6$	50	189	$1.43 \cdot 10^7$	$1.33 \cdot 10^7$	1.74	1.64
128	19	122	139	$4.10 \cdot 10^6$	$4.01 \cdot 10^6$	56	115	$7.19 \cdot 10^6$	$6.67 \cdot 10^6$	1.75	1.67
256	21	153	157	$2.03 \cdot 10^6$	$2.00 \cdot 10^6$	62	135	$3.49 \cdot 10^6$	$3.31 \cdot 10^6$	1.72	1.66

Table 4.1 compares AMR to LBG. The table displays the running times as well as the distortion values of the solutions attained by AMR and LBG for various n's. The running time is measured in Lloyd–Max iterations to obtain an implementation-independent comparison. Note that the computational cost of both AMR and LBG is dominated by that of the Lloyd–Max relaxations. The last column of Table 4.1 displays the ratio between the AMR distortion and the LBG distortion. It can be seen that, in this example, the first phase of the AMR algorithm (obtaining a guess) already yields a distortion that is close to the final result achieved by applying all three phases. It can also be seen that the complexity of AMR's first phase is $O(\log n)$

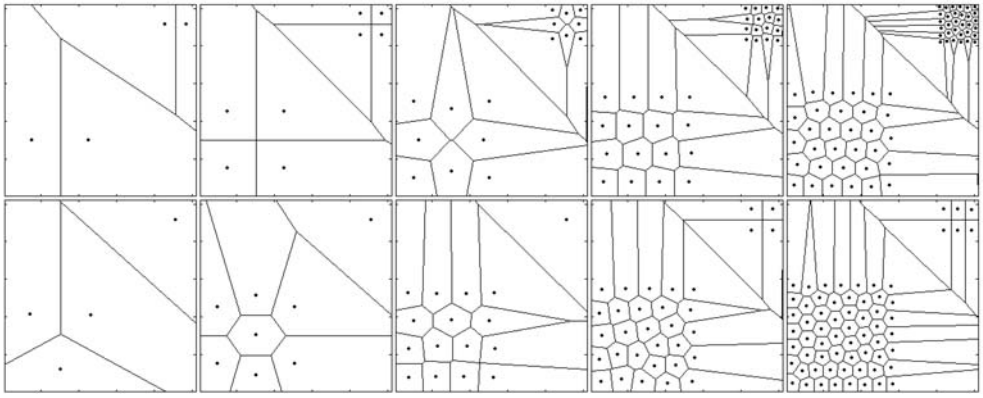

FIG. 4.3. *Quantization results of AMR and LBG applied to the p_{ij} that is defined by* (4.1) *and depicted in Figure* 4.2 *with* $n = 4, 8, 16, 32,$ *and* 64. *Top five plots: LBG. Bottom five plots: AMR.*

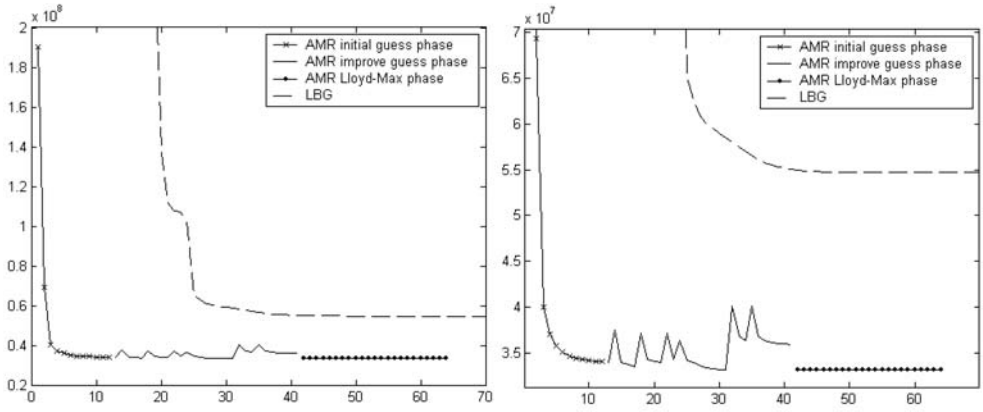

FIG. 4.4. *Distortion versus iterations for the AMR and LBG methods applied to p_{ij} that is defined in* (4.1) *and depicted in Figure* 4.2, *with* $n = 16$. *Note the different scale in each plot. Left: overall look at both processes. LBG's graph is dashed whereas AMR's three phases are solid with x-marks, solid without marks, and solid with point-marks, respectively. Right: close-up plot of the final distortion values.*

Lloyd–Max iterations, as multiplying n by 2 results in only a constant increase in the number of Lloyd–Max iterations.

From Figure 4.3 it is obvious that raising the LBG distortion improvement threshold for any $n > 2$ would not be beneficial. Raising the threshold would yield a solution with lower-valued distortion at each of the coarse scales. LBG implicitly assumes that coarse scale solutions are good initial guesses for the finer scales. Figure 4.3 demonstrates that, for example, when moving from $n = 2$ to $n = 4$, this is not necessarily the case.

In contrast to LBG, where the coarse scales lead us astray, AMR uses the coarse scales effectively to improve the fine scale solution. This is done by correcting errors at the coarse scales that were unrecognizable at the finer scales.

Figure 4.4 displays the distortion versus Lloyd–Max iterations for both LBG and

AMR when $n = 16$. The left-hand plot is an overall view of both processes. At its first stage, AMR begins with a uniform guess and redistributes the representation levels between the patches using the point density–based wealth function (3.3). Note the substantial drop in distortion achieved by the first three iterations. LBG falls short of this result even when run to completion. The phenomenon of 2–3 AMR iterations yielding lower distortion than tens or even hundreds of LBG iterations is quite common when quantizing sparse and patchy data. In general, for this kind of data, AMR may provide better solutions at a fraction of the computation time required by LBG, as LBG's solution at any time before $O(\log(n))$ Lloyd–Max iterations have been performed is quite poor. At its second phase, AMR tries to improve the solution by using (3.5). The purpose of this improvement stage is to traverse other feasible solutions that may have been overlooked at the first phase. Indeed, several improvements are found. Finally, the best solution to date is brought to a local minimum by application of Lloyd–Max iterations (1.5), (1.6). That is, Lloyd–Max iterations are performed until no further improvement is possible. The right-hand plot displays a close-up view of the processes when approaching the local minimum.

Our multiscale improvement method, $Improve-Guess$, was carried out until the first time that no transitions of representation levels were beneficial at any scale.

Table 4.1 demonstrates how the AMR solution may be significantly better than the LBG solution. We now show that the LBG solution may be arbitrarily bad.

OBSERVATION 1. *For any integer ℓ, there exist probability density functions $p(\vec{x})$, such that the distortion achieved by LBG when quantizing $p(\vec{x})$ is approximately ℓ times greater than the minimal distortion achievable.*

Proof. Consider a patchy $p(\vec{x})$. Let $p(\vec{x})$ be nonzero at only $\ell+1$ disjoint patches $\{\Omega_i\}_{i=1}^{\ell+1}$. One patch covers a considerable part of the domain Ω and the other ℓ patches cover an arbitrarily small part of Ω. Assume that $p(\vec{x})$ is constant within each patch and that $\int_{\Omega_i} p(\vec{x})dx$ is equal for all $1 \le i \le \ell+1$. That is, all patches have the same mass. Figure 4.5 is an example of such a $p(\vec{x})$ with eight patches that is defined formally in (4.2).

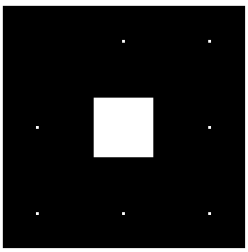

FIG. 4.5. *$p(x, y)$ with eight patches of different sizes but same mass, defined formally in* (4.2).

At the scale $n = \ell+1$, LBG's solution is, at best, comprised of one representation level per patch, which is the optimal solution. For larger n, LBG doubles each representation level. The Lloyd–Max relaxations do not permit representation levels to move from patch to patch, as the center of mass of any decision region always resides within the single patch covered by this region. Therefore, for any $n > \ell$, LBG places approximately $n/(\ell+1)$ representation levels in each patch. Assuming that the ℓ small patches are impulse functions (multiplied by the constant patch mass), the distortion that they contribute is zero once one or more representation levels are allocated to each of them. Therefore, for $n > \ell$, the optimal solution has $n-\ell$ representation levels

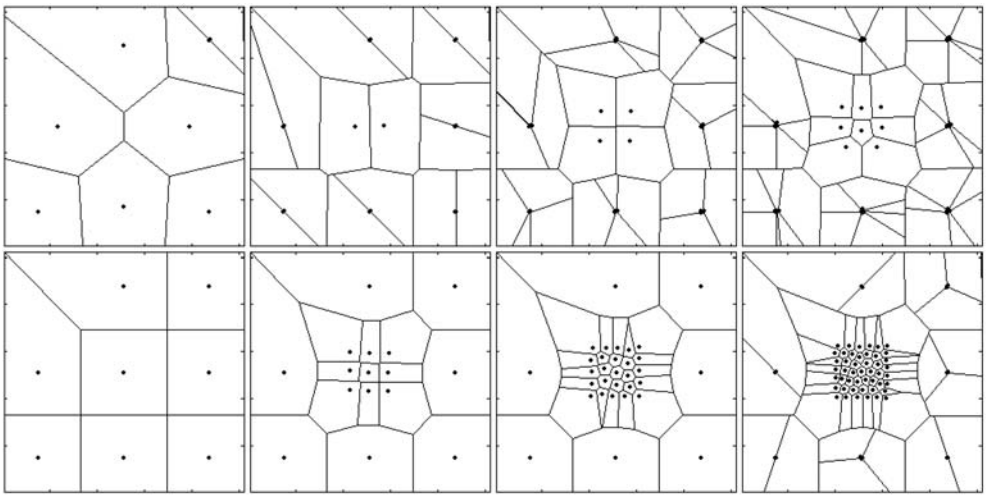

FIG. 4.6. *Quantization results of AMR and LBG on the eight-patch $p(\vec{x})$ defined by* (4.2) *and depicted in Figure* 4.5 *with* $n = 8, 16, 32,$ *and* 64. *Top four plots: LBG. Bottom four plots: AMR.*

in the large patch S, whereas LBG has only $n/(\ell+1)$ representation levels there.

It can be shown that, for the large patch, the distortion is roughly proportional to the inverse of the number of representation levels. This implies that the LBG solution is approximately ℓ times worse than the optimal solution.

More generally, for a d-dimensional problem, it can be shown that the distortion of the large patch is proportional to $n^{-2/d}$. Therefore, for a d-dimensional domain, one should choose a $p(\vec{x})$ with $(\ell+1)^{d/2}$ patches ($\ell+1$ in the 2-D case) to realize the stated result. □

Note that even when the ℓ smaller patches are not impulse functions, the optimal solution behaves similarly; that is, the great majority of representation levels are allocated to the larger patch. This is due to the quadratic penalty for input points that are far from their representative (1.1).

To demonstrate that Observation 1 is applicable in practice, consider the probability density function depicted in Figure 4.5. This $p(\vec{x})$ is comprised of one large patch and seven small ones, all of which have the same mass. $p(\vec{x})$ is formally defined as

$$
p(x,y) = \begin{cases}
1 & (x,y) \in \{97,98,\ldots,158,159\} \times \{97,98,\ldots,158,159\}, \\
441 & (x,y) \in \{36,37,38\} \times \{127,128,129\}, \\
441 & (x,y) \in \{36,37,38\} \times \{218,219,220\}, \\
441 & (x,y) \in \{127,128,129\} \times \{36,37,38\}, \\
441 & (x,y) \in \{127,128,129\} \times \{218,219,220\}, \\
441 & (x,y) \in \{218,219,220\} \times \{36,37,38\}, \\
441 & (x,y) \in \{218,219,220\} \times \{127,128,129\}, \\
441 & (x,y) \in \{218,219,220\} \times \{218,219,220\}, \\
0 & \textit{otherwise}.
\end{cases} \tag{4.2}
$$

Figure 4.6 presents the quantization results of AMR and LBG on this $p(\vec{x})$ when $n = 8, 16, 32,$ and 64. Note the differences in the number of representation levels per patch for various n in the LBG results. These differences are due to the aforementioned randomized implementation and the independence of each execution. Table 4.2

demonstrates how the AMR solution for this $p(\vec{x})$ is generally between four and six times better than the LBG solution for various n. In general, the larger the number of patches and the smaller the area that each of the smaller patches covers, the worse the LBG solution becomes relative to the optimal solution and the AMR solution. Note that the computation time of the LBG algorithm in this example is quite fast. This is because the Lloyd–Max relaxations mainly affect the representation levels in the large patch. As the Lloyd–Max process's convergence rate is highly dependent on the number of representation levels, and as only a fraction of the original number of representation levels n are active, the LBG algorithm converges faster than for a more general $p(\vec{x})$. The gain of AMR over LBG is not as high as predicted by Observation 1 only because the patches are not impulse functions.

TABLE 4.2

Results of LBG and AMR applied to the eight-patch $p(\vec{x})$ defined by (4.2) *and depicted in Figure* 4.5. *This table's structure is identical to that of Table* 4.1.

n	**AMR iter.**			**AMR dist.**		**LBG iter.**		**LBG dist.**		**LBG:AMR**	
8	10	56	57	$5.00 \cdot 10^7$	$2.66 \cdot 10^6$	11	24	$1.66 \cdot 10^7$	$1.64 \cdot 10^7$	0.33	6.16
16	13	56	94	$4.27 \cdot 10^5$	$3.28 \cdot 10^5$	14	21	$1.69 \cdot 10^6$	$1.66 \cdot 10^6$	3.96	5.06
32	14	32	42	$1.47 \cdot 10^5$	$1.41 \cdot 10^5$	17	32	$6.76 \cdot 10^5$	$6.72 \cdot 10^5$	4.60	4.77
64	16	72	96	$7.78 \cdot 10^4$	$7.60 \cdot 10^4$	22	73	$3.88 \cdot 10^5$	$3.50 \cdot 10^5$	4.99	4.60

5. Conclusions and further discussion. With respect to the multiple local minima problem, we have demonstrated an adaptive multiscale redistribution (AMR) strategy based on the point density function and distortion attributes of an optimal quantizer.[1] This strategy is nonlocal and may determine from a local glance that the solution is not globally optimal, an attribute not shared by the Lloyd–Max method. Our strategy includes a fast method for obtaining an initial guess, as well as a method for improving a guess and bringing it to a local minimum. This strategy may beat LBG by an arbitrary factor, and it may be substantially faster than LBG.

In a previous paper [20] we have presented a multigrid acceleration of the 1-D Lloyd–Max iterative process. That is, given an initial guess, a local minimum near it may be reached at a complexity equivalent to just a few Lloyd–Max iterations. Using the AMR methods for the scalar case may be very powerful. This is because, at the guess improvement stage, we may replace every application of Lloyd–Max iterations (step 2(b)(i) of $Improve-Guess$) with our monotonically convergent multigrid solver [20]. Doing so implies that we need not guess if a transition of a representation level is worthwhile. Instead, we can use our multigrid method to arrive at a local minimum and compare its distortion with the best distortion to date.

An interesting question is whether or not an acceleration scheme similar to that of the scalar quantization problem exists for vector quantization. We believe that a standard multigrid approach, such as the one in [20], cannot work since the problem does not behave "elliptically." A small change in the solution at some point may cause a large change in the solution at some other distant point. For this reason, the Lloyd–Max relaxations do not smooth the distortion/residual, which is a basic demand of multigrid schemes.

We believe that the ideas proposed in this paper may be applied to other interesting representation problems, for example, representing a k-dimensional manifold $S(\vec{x})$ by n points. From these n points, given some interpolation scheme, one can

[1] Implementations are available at http://www.cs.technion.ac.il/~irad/Quantization/index.htm.

compute an approximating manifold $S'(\vec{x})$. The objective is to minimize

$$\int \|S(\vec{x}) - S'(\vec{x})\|^2 d\vec{x}. \tag{5.1}$$

With suitable adaptations, our proposed methods may also be able to deal with other image processing problems such as halftoning, where the average color value of the halftoned image should equal the average color value of the original image at all scales. These problems are currently being researched.

Appendix. The importance of *multiscale* redistribution. Performing redistribution on the finest scale alone yields noteworthy results. Nevertheless, wrapping the redistribution operation within a multiscale framework yields a noticeably better performance, as the following observation implies.

OBSERVATION 2. *There are examples for which the redistribution process can occur only at coarse scales. Moreover, at any coarse scale, the number of representation levels redistributed may be as high as half of the number of aggregates of decision regions at that scale, which reduces the distortion significantly.*

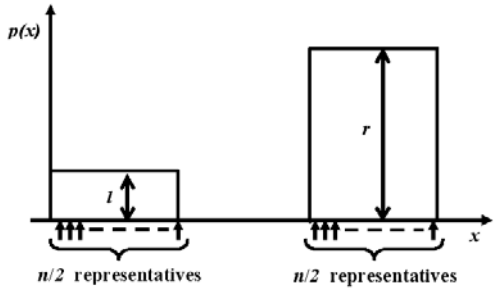

FIG. A.1. *Probability density function $p(x)$ with two patches.*

Proof. Consider the probability density function

$$p(x) = \begin{cases} l : 0 \le x \le 1/3, \\ r : 2/3 \le x \le 1, \\ 0 : otherwise. \end{cases} \tag{A.1}$$

For simplicity, we choose a 1-D $p(x)$. Similar examples exist for any dimension. Assume that $n/2$ representation levels are allocated to each of the two patches of $p(x)$. This configuration is depicted in Figure A.1. For any l and r, uniformly distributing the representation levels in each patch yields a local minimum of the distortion. For any such local minimum, in each patch separately, the wealth of any two decision regions is equal since $p(x)$ is constant. Denote by $\mathcal{W}_l$ and $\mathcal{W}_r$ the wealth of any decision region in the left and right patches, respectively. Evidently, there exist values of l and r such that $\mathcal{W}_l = T/2 - \epsilon$ and $\mathcal{W}_r = -T/2 + \epsilon$, where T is either the chosen threshold in the $Improve - Guess$ algorithm or $T = 1$ for the $Obtain - Guess$ algorithm. Recall that representation levels are redistributed if and only if $\mathcal{W}_l - \mathcal{W}_r > T$. At the finest scale, no redistribution occurs since $\mathcal{W}_l - \mathcal{W}_r = T - 2\epsilon$. At the second finest scale, where Ω is partitioned into, say, $n/2$ aggregates of decision regions, $\mathcal{W}_l = T - 2\epsilon$ and $\mathcal{W}_r = -T + 2\epsilon$, which implies $\mathcal{W}_l - \mathcal{W}_r > T$ for small enough ϵ, and therefore redistribution occurs. In fact, $n/4$ representation levels are transferred from

the left patch of $p(x)$ to the right patch at this scale. For general scale m, where Ω is partitioned into m aggregates of decision regions after approximately $\log_2(m)$ aggregations, the proof is similar. There exist values of l and r such that, at the finest scale, $\mathcal{W}_l = T/(n/m) - \epsilon$ and $\mathcal{W}_r = -T/(n/m) + \epsilon$. All scales finer than m fail to comply with the redistribution criterion $\mathcal{W}_l - \mathcal{W}_r > T$. However, at scale m, for small enough ϵ, the criterion is satisfied and redistribution occurs because the wealth per m-scale aggregate, which consists of n/m representation levels, is equal to n/m times the fine scale wealth. In fact, approximately $m/2$ representation levels are transferred from the left patch of $p(x)$ to the right patch at this scale. □

REFERENCES

[1] R. Balasubramanian, C. A. Bouman, and J. P. Allebach, *Sequential scalar quantization of vectors: An analysis*, IEEE Trans. Image Process., 4 (1995), pp. 1282–1295.

[2] W. R. Bennett, *Spectra of quantized signals*, Bell Systems Tech. J., 27 (1948), pp. 446–472.

[3] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31 (1977), pp. 333–390.

[4] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed., SIAM, Philadelphia, 2000.

[5] R. Clark, *An extended dissimilarity selection method for finding diverse representative subsets*, J. Chem. Inform. Comput. Sci., 37 (1997), pp. 1181–1188.

[6] J. H. Conway and N. J. A. Sloane, *Sphere Packings, Lattices and Groups*, Springer-Verlag, New York, 1988.

[7] Q. Du, V. Faber, and M. Gunzburger, *Centroidal Voronoi tesselations: Applications and algorithms*, SIAM Rev., 41 (1999), pp. 637–676.

[8] P. Fleischer, *Sufficient conditions for achieving minimum distortion in a quantizer*, IEEE Internat. Convention Record, (1964), pp. 104–111.

[9] J. H. Freidman, J. L. Bentley, and R. A. Finkel, *An algorithm for finding best matches in logarithmic expected time*, ACM Trans. Math. Software, 3 (1977), pp. 209–226.

[10] A. Gersho, *Asymptotically optimal block quantization*, IEEE Trans. Inform. Theory, 25 (1979), pp. 373–380.

[11] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1992.

[12] R. M. Gray and E. D. Karnin, *Multiple local optima in vector quantizers*, IEEE Trans. Inform. Theory, 28 (1982), pp. 256–261.

[13] R. M. Gray and D. L. Neuhoff, *Quantization*, IEEE Trans. Inform. Theory, 44 (1998), pp. 1–63.

[14] J. Hartigan, *Clustering Algorithms*, Wiley-Interscience, New York, 1975.

[15] S. Hasegawa, H. Imai, M. Inaba, N. Katoh, and J. Nakano, *Efficient algorithms for variance-based k-clustering*, in Proceedings of the First Pacific Conference on Computer Graphics and Applications, IEEE, Piscataway, NJ, 1993, pp. 75–89.

[16] P. Heckbert, *Color image quantization for frame buffer display*, in Proceedings of the 9th Annual Conference on Computer Graphics and Interactive Images, ACM, New York, 1982, pp. 297–307.

[17] H. Honda, *Description of cellular patterns by Dirichlet domains: The two-dimensional case*, J. Theoret. Biol., 72 (1978), pp. 523–543.

[18] H. Imai, N. Katoh, and M. Inaba, *Applications of weighted voronoi diagrams and randomization to variance-based k-clusteringgeometric k-clustering*, in Proceedings of the 10th ACM Symposium on Computational Geometry, ACM, New York, 1994, pp. 332–339.

[19] A. Jain and R. Dubes, *Algorithms for Clustering Data*, Prentice–Hall, Englewood Cliffs, NJ, 1998.

[20] Y. Koren, I. Yavneh, and A. Spira, *A multigrid approach to the 1-D quantization problem*, IEEE Trans. Inform. Theory, to appear.

[21] Y. Linde, A. Buzo, and R. M. Gray, *An algorithm for vector quantizer design*, IEEE Trans. Commun., 28 (1980), pp. 84–95.

[22] S. P. Lloyd, *Least squares quantization in PCM*, IEEE Trans. Inform. Theory, 28 (1982), pp. 129–137.

[23] J. MacQueen, *Some methods for classification and analysis of multivariate observations*, in Proceedings of the Fifth Berkeley Symposium on Mathematics, Statistics, and Probability, Vol. 1: Statistics, University of California Press, Berkeley, CA, 1967, pp. 281–296.

[24] J. Max, *Quantizing for minimal distortion*, IEEE Trans. Inform. Theory, 6 (1960), pp. 7–12.
[25] M. T. Orchard and C. A. Bouman, *Color quantization of images*, IEEE Trans. Signal Process., 39 (1991), pp. 2677–2690.
[26] P. F. Panter and W. Dite, *Quantizing distortion in pulse-count modulation with nonuniform spacing of levels*, in Proc. IRE, 39, 1951, pp. 44–48.
[27] L. Torres and J. Huguet, *An improvement on codebook search for vector quantization*, IEEE Trans. Commun., 42 (1994), pp. 208–210.
[28] L. F. Toth, *Sur la representation d'une population infinie par un nombre fini d'elements*, Acta Math. Acad. Sci. Hungar., 25 (1959), pp. 76–81.
[29] U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, New York, 2001.
[30] S. J. Wan and S. K. M. Wong, *An algorithm for multidimensional data clustering*, ACM Trans. Math. Software, 14 (1988), pp. 153–162.
[31] X. Wu, *Color quantization by dynamic programming and principal analysis*, ACM Trans. Graphics, 11 (1992), pp. 348–372.
[32] X. Wu and L. Guan, *Acceleration of the LBG algorithm*, IEEE Trans. Commun., 42 (1994), pp. 1518–1523.
[33] P. L. Zador, *Development and Evaluation of Procedures for Quantizing Multivariate Distributions*, Ph.D. thesis, Stanford University, Stanford, CA, 1963. Also Stanford University Department of Statistics Technical Report.