Image Reduction Using Means on Discrete Product Lattices

Gleb Beliakov Senior Member, IEEE, Humbeto Bustince Member, IEEE, and Daniel Paternain

Abstract—We investigate the problem of averaging values on lattices, and in particular on discrete product lattices. This problem arises in image processing when several color values given in RGB, HSL, or another coding scheme, need to be combined. We show how the arithmetic mean and the median can be constructed by minimizing appropriate penalties, and we discuss which of them coincide with the Cartesian product of the standard mean and median. We apply these functions in image processing. We present three algorithms for color image reduction based on minimizing penalty functions on discrete product lattices.

Index Terms—Aggregation Operators, Penalty Functions, Image reduction, Mean, Median.

I. INTRODUCTION

The need to aggregate several inputs into a single representative output frequently arises in many practical applications. In image processing, it is often necessary to average the values of several neighboring pixels (to reduce the image size or apply a filter), or average pixel values in two different but related images (e.g., in stereovision [1]). When the images are in color, typically coded as discrete RGB, CMY, or HSL values, then it is customary to average the values in the respective channels. It is not immediately clear that this is appropriate, and what are the other ways to average color values.

In this paper we study averaging on product lattices (RGB or another color coding scheme is an example of a product lattice). We note previous works related to triangular norms on posets and lattices [2], [3] and on discrete chains [4]. Our setting is different as we do not deal with associativity of aggregation operations, but in contrast require averaging behavior.

We focus on a large class of averages based on minimizing a penalty function [5]–[8]. We show that with an appropriately chosen class of penalties, the resulting penalty-based functions are monotone and idempotent. We also show that the averages over a product lattice are in general different from the Cartesian product of the averages. This has an implication over the methods of color image reduction.

We recall the problem of image reduction for grayscale images and we justify the importance of penalty functions.

II. PRELIMINARIES

A. Aggregation functions

The research effort concerning aggregation functions, their behavior and properties, has been disseminated throughout various fields including decision making, knowledge based systems, artificial intelligence and image processing. Recent works providing a comprehensive overview include [9]–[13].

Definition 1: A function \( f : [a,b]^n \to [a,b] \) is called an aggregation function if it is monotone non-decreasing in each variable and satisfies \( f(a) = a, f(b) = b \), with \( a = (a,a,\ldots,a), b = (b,b,\ldots,b) \).

Definition 2: An aggregation function \( f \) is called averaging if it is bounded by the minimum and maximum of its arguments

\[
\min(x) := \min(x_1, \ldots, x_n) \leq f(x_1, \ldots, x_n) \leq \max(x_1, \ldots, x_n) =: \max(x).
\]

It is immediate that averaging aggregation functions are idempotent (i.e., \( \forall t \in [a,b] : f(t, t, \ldots, t) = t \)) and (because of monotonicity) vice versa. Then clearly the boundary conditions \( f(a) = a, f(b) = b \) are satisfied.

Well known examples of averaging functions are the arithmetic mean and the median. It is known that the arithmetic mean and the median are solutions to simple optimization problems, in which a measure of disagreement between the inputs is minimized, see [5]–[7], [10], [14]. The main motivation is the following. Let \( x \) be the inputs and \( y \) be the output. If all the inputs coincide \( x = x_1 = \ldots = x_n \), then the output is \( y = x \), and we have a unanimous vote. If some input
$x_i \neq y$, then we impose a "penalty" for this disagreement. The larger the disagreement, and the more inputs disagree with the output, the larger (in general) is the penalty. We look for an aggregated value which minimizes the penalty. Thus we need to define a suitable measure of disagreement, or dissimilarity.

**Definition 3:** Let $P : [a, b] \rightarrow \mathbb{R}$ be a penalty function with the properties

i) $P(x, y) \geq 0$ for all $x, y$;

ii) $P(x, y) = 0$ if all $x_i = y$;

iii) $P(x, y)$ is quasiconvex in $y$ for any $x$.

The penalty based function is

$$f(x) = \arg \min_y P(x, y),$$

if $y$ is the unique minimizer, and $y = \frac{a+b}{2}$ if the set of minimizers is the interval $[a, b]$.

**Remark 1:** $f$ is quasiconvex if $f(ax_1 + (1-a)y_2) \leq \max\{f(y_1), f(y_2)\}$ for all $a \in [0, 1]$ and all $y_1, y_2$ within its domain.

In [5] it was shown that any averaging aggregation function can be represented as a penalty based function. Further, the classical means, such as the arithmetic mean and the median are represented via the following penalty functions. The arithmetic mean is the solution to

$$\min_y \sum_{i=1}^n (x_i - y)^2$$

whereas the median is a solution to

$$\min_y \sum_{i=1}^n |x_i - y|.$$
do not consider this problem. We focus on the influence of the choice of the measure of error in the second point.

For simplicity, we consider the following reconstruction method: for each pixel of the reduced image build a new block of dimension $n \times n$ whose elements have the same value as that pixel.

Next we show that once the reduction and magnification methods are fixed, the difference between the original and the reduced (and then magnified) image may be determined by the aggregation function used in the reduction algorithm.

We measure the error in the reconstructed images by using the following expressions to compare two images $Q, Q'$ of dimension $N \times M$: Mean Squared Error (MSE) and Mean Absolute Error (MAE)

$$MSE(Q, Q') = \frac{1}{N \times M} \sum_{i=1}^{N} \sum_{j=1}^{M} (Q_{ij} - Q'_{ij})^2$$
$$MAE(Q, Q') = \frac{1}{N \times M} \sum_{i=1}^{N} \sum_{j=1}^{M} |Q_{ij} - Q'_{ij}|$$

Notice that from the results in Table I we have that:

1) If we take MSE, then the best reduction is obtained using the arithmetic mean.

2) If we take MAE, then the best reduction is obtained using the median.

Observe that these two facts agree with Section II-A and justify the study of penalty functions for image reduction. We are interested in color images, and hence our interest to penalty functions defined over the product lattices.

III. MAIN DEFINITIONS

Following representations of the arithmetic mean and the median as penalty based aggregation functions, we now define similar constructions on lattices.

Definition 8: Let $L = (L, \leq, \land, \lor)$ be a product of finite chains. The distance between $x, y \in L$ is defined as the length of a maximal chain $C$ with the least element $a = x \land y$ and the greatest element $b = x \lor y$ minus 1.

$$d(x, y) = \text{length}(C) - 1.$$  

This distance is called the geodesic distance, since it corresponds to the smallest number of edges between vertices $x$ to $y$ in the covering graph of $L$.

Remark 2: We note that all maximal chains with the least element $a$ and the greatest element $b$ on a product lattice in Definition 8 have all the same length. This definition is equivalent to the following

$$d(x, y) = \sum_{i=1}^{m} d_i(x_i, y_i) = \sum_{i=1}^{m} |x_i - y_i|,$$

where $d_i$ is the distance in the $i$-th chain in the product of $m$ chains.

Definition 9: Let $L$ be a product of finite chains. Consider $n$ elements $x_1, \ldots, x_n \in L$, that need to be averaged. Let the penalty function be $P : Z^n_+ \to \mathbb{R}$. The penalty based function on $L$ is $f$ given by

$$f(x_1, \ldots, x_n) = \mu = \arg\min_{y \in L} P(d(x_1, y), d(x_2, y), \ldots, d(x_n, y)).$$

Remark 3: $P$ is quasi-convex in $y$, as in Definition 3. But now $y \in L$ rather than an element of a chain. To accommodate this in the definition of a quasi-convex function we use the following. We remind that a function $f : X \subseteq \mathbb{R}^n \to R$ is quasi-convex on $X$ if all its level sets are convex. We call the function $f : L \subseteq Z^n_+ \to \mathbb{R}$ quasi-convex if its extension $f : X \to \mathbb{R}$ is quasi-convex, where $X \subseteq \mathbb{R}^n$ is the smallest convex set containing $L$. Similarly $f : L \to \mathbb{R}$ is convex if its extension $f$ is convex.

The minimum always exists and $\mu \in L$. There can be several minimizers. In this case one can take any minimizer. A convenient rule is to take the largest minimizer according to

<table>
<thead>
<tr>
<th>Image (b1)</th>
<th>Image (b2)</th>
<th>Image (b3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE</td>
<td>224.0065</td>
<td>209.23</td>
</tr>
<tr>
<td>MAE</td>
<td>7.89</td>
<td>7.88</td>
</tr>
</tbody>
</table>

TABLE I

MSE AND MAE BETWEEN RECONSTRUCTED IMAGES OF FIG. 2 AND ORIGINAL LENA IMAGE

![Fig. 1. Scheme of the reduction algorithm](image1)

![Fig. 2. Original image Lena (a) and reductions by grayscale image reduction algorithm taking $n = 3$ and (b) geometric mean, (c) arithmetic mean and (d) median](image2)
a total order, e.g., lexicographical. Finally, $f$ is not necessarily monotone, i.e., an aggregation function.

**Theorem 1:** The function $f$ in Definition 9, is an averaging (and hence idempotent) function.

Proof: Clearly $\bigwedge x_i = a \leq \mu \leq b = \bigvee x_i$, because for any $x_i, d(x_i, a) < d(x_i, t)$ with $t < a$, and similarly at the other end.

A special case of penalty based functions was considered in [8], called dissimilarity functions (see also [23], [24]), where the penalty $P$ is given by

$$P(x, y) = \sum_{i=1}^{n} K(x_i - y),$$

(1)

where $K$ is a convex function with the unique minimum $K(0) = 0$. In this case the penalty based function is monotone, i.e., an aggregation function. By adapting this definition to our case we have the following result:

**Theorem 2:** The function $f$ in Definition 9, with $P$ given by

$$P(x, y) = \sum_{i=1}^{n} K(d(x_i, y)),$$

is an averaging aggregation function on a product lattice.

Proof: We only need to prove monotonicity, the proof is similar to that in [8], see also [5], and is adapted here to product lattices. A convex function $K$, increasing on $[0, \infty)$ has the property $K(u) - K(v) \leq K(u') - K(v')$ if $u - v \leq u' - v'$ and $u, v, u', v' \geq 0, u' > u$. Consider $x$ and $x'$, such that $x'_i = x_i$ for all $i, j$ except one pair, $x'_k = x_k + 1$. We need to show that if $P(x, y^*) \leq P(x, y)$ for all $y \in \mathcal{L}$, then $P(x', y^*) < P(x', y^*)$ for all $y < y^*$.

Suppose $y < y^*$. Take $u = \sum_{j=1}^{m} = \sum_{j \neq i} x_j - y_j^* + |x_k + 1 - y_k^*|, v = \sum_{j=1}^{m} |x_k - y_j^*|, u' = \sum_{j \neq i} |x_k - y_j^*| + |x_k + 1 - y_k^*|, v' = \sum_{j=1}^{m} |x_k - y_j^*|$. Now, $u - v = |x_k + 1 - y_k^*| - |x_k - y_k^*|$, which is $-1$ if $y_k^* \leq x_k$ or $1$ otherwise. Also $u' - v' = |x_k + 1 - y_k^*| - |x_k - y_k^*|$, which is also either $+1$ or $-1$, but because $y < y^*$, it takes value $-1$ only if $u - v = -1$. Hence $u - v = u' - v'$.

Now from $P(x, y^*) \leq P(x, y)$ we have

$$P(x', y^*) = \sum_{i \neq k} K(d(x_i, y^*)) + K(u) \leq \sum_{i \neq k} K(d(x_i, y)) + K(u') = P(x', y).$$

**Remark 4:** One can use $n$ distinct convex functions $K_i, i = 1, \ldots, n$ in (1) rather than a common function $K$, and the result of Theorem 2 holds. In particular, an interesting case is $K_i(x) = w_i K(x), with w_i \geq 0, \sum w_i = 1$, which gives rise to weighted means and medians.

Below we provide definitions for some specific instances of penalty based aggregation, based on the analogs with the classical means. In all cases we have penalties in the form (1), so Theorem 2 applies.

**Definition 10:** Let $K$ be:

1) $K(x) = x^2$, and hence $P(x, y) = \sum_{i=1}^{n} d(x_i, y)^2$. Then the resulting penalty based aggregation function is the arithmetic mean.

2) $K_i(x) = w_i x^2$, and hence $P(x, y) = \sum_{i=1}^{n} w_i d(x_i, y)^2$, and $w$ be a weighting vector, $w_i \geq 0, \sum w_i = 1$. Then the resulting penalty based aggregation function is a weighted arithmetic mean.

3) $K(x) = |x|$, and hence $P(x, y) = \sum_{i=1}^{n} d(x_i, y)$. Then the resulting penalty based aggregation function is the median.

4) $K_i(x) = w_i |x|$, and hence $P(x, y) = \sum_{i=1}^{n} w_i d(x_i, y)$. Then the resulting penalty based aggregation function is a weighted median (the definitions of the weighted medians can be found in [5], [6], [9]).

**IV. Solution to Penalty Minimization Problems**

Consider now the issue of obtaining solutions to the minimization problem in Definition 9. First, consider the arithmetic mean. We have the problem

$$\text{minimize}_{y \in \mathcal{L}} \sum_{i=1}^{n} \left( \sum_{j=1}^{m} |x_{ij} - y_j| \right)^2,$$

(2)

where $x_{ij}$ denotes the $j$-th component of the $i$-th tuple $x_i \in \mathcal{L}$. We note that this problem is convex in $y$. We also note that the solution is different from the Cartesian product of the means, as the following example illustrates, and the differences are not just due to the rounding problem.

**Example 1:** Let $\mathcal{L}$ be the product of two chains $0, \ldots, 10$. Take the mean of $(10, 10), (8, 0), (3, 2)$. The Cartesian product of means gives $(7, 4)$, with the objective value $9^2 + 5^2 + 6^2 = 142$. The solutions to the minimization problem are $(9, 2)$ with the objective $9^2 + 3^2 + 6^2 = 126$ and $(8, 3)$ with the same objective.

While we could not obtain a closed form solution, we note that starting from any $y \in \mathcal{L}$, and in particular starting from the Cartesian product of means or the medians, and performing coordinate descent (because of the convexity of the objective), one can reach the minimum algorithmically.

Consider now the median. For the median we have the problem

$$\text{minimize}_{y} \sum_{i=1}^{n} \sum_{j=1}^{m} |x_{ij} - y_j| = \sum_{j=1}^{m} \sum_{i=1}^{n} |x_{ij} - y_j|.$$

(3)

Each term in the inner sum in the latter expression depends on $y_j$ only, thus the solution to the problem can be obtained by solving $m$ separate problems

$$\min_{y_j} \sum_{i=1}^{n} |x_{ij} - y_j|.$$

The solution to each of these problems is the median function. Hence the minimum in (3) is achieved at $y = (\text{Med}_x(x_1), \ldots, \text{Med}_x(x_m))$, i.e., the result is the Cartesian product of the medians. It is not difficult to confirm that the same conclusion is also valid for weighted medians.

There are several interesting examples of penalty functions presented in [5], which give rise to their analogues defined on product lattices. None of them results in a Cartesian product of the respective aggregation functions though.
V. APPLICATION IN IMAGE REDUCTION

In this section we consider a practical application of aggregation on product lattices to color image processing. We present three color image reduction algorithms based on minimization of penalty functions that are not built as Cartesian products of the corresponding aggregation functions. The first two algorithms are approximate algorithms: they provide putative solutions to the penalty minimization problem, chosen from smaller subsets of alternatives. The rationale here is computational efficiency. The third algorithm finds the actual solution to the penalty minimization problem using the approach in Section IV. We compare the accuracy and running times for all algorithms.

A. First algorithm for image reduction

1) The algorithm: In the first color image reduction algorithm we fix a number of $k$ different averaging aggregation functions. We apply the aggregation functions to each of the blocks in the image (componentwise) obtaining $k$ possible pixels in the reduced image. We select the pixel that minimizes a fixed penalty function $P$. A diagram of Algorithm 1 can be found in Fig. 3.

Algorithm 1 First color image reduction algorithm

Input: $Q$ of dimension $N \times M$

Output: $Q'$ of dimension $\frac{N}{T} \times \frac{M}{T}$

1. Divide the image $Q$ in disjoint blocks of dimension $n \times n$.
   If $N$ or $M$ are not multiples of $n$ eliminate the necessary number of rows and/or columns to satisfy this condition.
2. Choose a penalty function $P$.
3. Take $k$ averaging aggregation functions $A_{g_1}, \ldots, A_{g_k}$.
4. for each block $x$ in $Q$ do
5. Apply to each pixel in each block (in the three channels R, G and B) $k$ aggregation functions, as follows:

   $y_{A_{g_1}} = (y_{RA_{g_1}}, y_{GA_{g_1}}, y_{BA_{g_1}}) =$
   $= (A_{g_1}(x_{Rij}), A_{g_1}(x_{Gij}), A_{g_1}(x_{Bij}))$
   $\cdots$

   $y_{A_{g_k}} = (y_{RA_{g_k}}, y_{GA_{g_k}}, y_{BA_{g_k}}) =$
   $= (A_{g_k}(x_{Rij}), A_{g_k}(x_{Gij}), A_{g_k}(x_{Bij}))$

6. Calculate the penalties $P_i = P(x, y_{A_{g_i}})$ for each $y_{A_{g_i}}$ with $i = 1, \ldots, k$.
7. Assign the value $y_{A_{g_i}}$ with the smallest penalty to the corresponding pixel of the reduced image.

$\begin{align*}
\arg \min_{y_{A_{g_i}}} P(x, y_{A_{g_i}})
\end{align*}$

8. end for

We illustrate the Algorithm 1 on the following example. We reduce a block of dimension $3 \times 3$. We take 5 different aggregation functions: minimum (min), geometric mean (geom), arithmetic mean (arith), median (med) and maximum (max).

Example 2: We consider the following block of an image:

<table>
<thead>
<tr>
<th>R</th>
<th>G</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>115</td>
<td>100</td>
<td>120</td>
</tr>
<tr>
<td>130</td>
<td>125</td>
<td>125</td>
</tr>
<tr>
<td>135</td>
<td>130</td>
<td>130</td>
</tr>
<tr>
<td>221</td>
<td>220</td>
<td>220</td>
</tr>
<tr>
<td>150</td>
<td>200</td>
<td>210</td>
</tr>
<tr>
<td>215</td>
<td>200</td>
<td>215</td>
</tr>
</tbody>
</table>

Suppose the following penalty function (corresponding to the arithmetic mean) is fixed

$P(x, y) = \sum_{i=1}^{3} \sum_{j=1}^{3} (\sum_{C \in \{R, G, B\}} |x_{Cij} - y_{C}|)^2$

We apply the aggregation functions to the elements of the block componentwise obtaining the following results:

$y_{min} = (10, 100, 150)$, $P(x, y_{min}) = 62520$
$y_{geom} = (11.98, 122.9, 204.41)$, $P(x, y_{geom}) = 7904$
$y_{arith} = (12.11, 123.33, 205.67)$, $P(x, y_{arith}) = 7714$
$y_{med} = (12, 125, 215)$, $P(x, y_{med}) = 7364$
$y_{max} = (15, 135, 220)$, $P(x, y_{max}) = 10920$

For this block of the image we take $y_{med}$, which corresponds to the smallest value of the penalty $P$ for this block. Note that even though the penalty function corresponds to the arithmetic mean, the solution is not $y_{arith}$ (which corresponds to the Cartesian product of arithmetic means), which is consistent with the argument in Section IV.

In Fig. 4 we illustrate Algorithm 1 on two color images in RGB (images (a) and (c)) in the same setting as in the Example 2. In Table II we show the frequency of choosing each of the aggregation functions. Notice that the biggest percentage corresponds to taking the arithmetic mean as aggregation function.

Remark 5: Observe that if all the values of the color components are the same, we can take any averaging function, because they are all idempotent (column Any in Table II).

<table>
<thead>
<tr>
<th></th>
<th>Min</th>
<th>Geom</th>
<th>Arith</th>
<th>Med</th>
<th>Max</th>
<th>Any</th>
</tr>
</thead>
<tbody>
<tr>
<td>Im (a)</td>
<td>0.03%</td>
<td>6.14%</td>
<td>63.08%</td>
<td>26.09%</td>
<td>0.02%</td>
<td>4.04%</td>
</tr>
<tr>
<td>Im (c)</td>
<td>0.03%</td>
<td>5.43%</td>
<td>70.50%</td>
<td>17.72%</td>
<td>0.05%</td>
<td>6.21%</td>
</tr>
</tbody>
</table>
2) Reaction to noise: Now we want to analyze how Algorithm 1 behaves when images have been altered with impulsive noise of the salt and pepper type, frequent in practice. We modify the images in Fig. 4 adding a 5, 10, 20 and 30% of noise density. In Table III we show the frequency of choosing each aggregation function in the setting of Example 2. The results are illustrated in Fig. 5. In first row we show original images with noise. In second row we show the images obtained applying Algorithm 1 and a simple reduction algorithm of subsampling.

Notice that when the amount of salt and pepper noise in the images increases, the frequency of choosing the median also increases. This can be seen in Fig. 6. On the horizontal axis we show the percentage of pixels affected by noise. On the vertical axis we show the percentage of times that each aggregation function is selected by Algorithm 1. The larger is impulsive noise, the more often the median is selected instead of the arithmetic mean.

As the median is taken most frequently over each block of the image, Algorithm 1 allows one to discard the impulsive noise. This is explained by the fact that the median is not affected by the extremal values that are taken by the corrupted pixels.

The main advantage of Algorithm 1 is that it makes unnecessary to use an ad-hoc filter prior to the image reduction in order to eliminate this kind of noise.

![Fig. 4. Original color images (a) and (c) and reduced images (b) and (d) applying Algorithm 1](image)

![Fig. 5. Original images with 20% of impulsive noise (a) and (c) and reductions applying Algorithm 1 (b1) and (d1) and subsampling algorithm (b2) and (d2).](image)

<table>
<thead>
<tr>
<th>Image and noise density</th>
<th>Min</th>
<th>Geom</th>
<th>Arith</th>
<th>Med</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Im (a) 5%</td>
<td>0.15%</td>
<td>6.90%</td>
<td>62.76%</td>
<td>30.13%</td>
<td>0.06%</td>
</tr>
<tr>
<td>Im (c) 5%</td>
<td>0.14%</td>
<td>8.25%</td>
<td>61.09%</td>
<td>30.42%</td>
<td>0.10%</td>
</tr>
<tr>
<td>Im (a) 10%</td>
<td>0.20%</td>
<td>5.44%</td>
<td>31.17%</td>
<td>43.12%</td>
<td>0.07%</td>
</tr>
<tr>
<td>Im (c) 10%</td>
<td>0.15%</td>
<td>5.76%</td>
<td>46.13%</td>
<td>47.86%</td>
<td>0.10%</td>
</tr>
<tr>
<td>Im (a) 20%</td>
<td>0.14%</td>
<td>2.60%</td>
<td>34.87%</td>
<td>62.37%</td>
<td>0.02%</td>
</tr>
<tr>
<td>Im (c) 20%</td>
<td>0.14%</td>
<td>1.97%</td>
<td>28.41%</td>
<td>69.42%</td>
<td>0.06%</td>
</tr>
<tr>
<td>Im (a) 30%</td>
<td>0.09%</td>
<td>1.55%</td>
<td>27.55%</td>
<td>70.81%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Im (c) 30%</td>
<td>0.06%</td>
<td>0.73%</td>
<td>24.99%</td>
<td>74.19%</td>
<td>0.03%</td>
</tr>
</tbody>
</table>

TABLE III

Frequency of choosing aggregation functions by Algorithm 1 when images (a) and (c) of Fig. 4 are affected by salt and pepper noise.
Example 2. The first image (image (a)) is a synthetic one with Algorithm 2 of Algorithm 1. In these cases, the value of the
same penalty conditions solutions of Algorithm 2 are different from those of possible outputs of Algorithm 2. We analyze under which
k of all the possible combinations of P

Second algorithm for image reduction

We see that Algorithm 1 does not ensure that we select the global minimizer of the penalty function P by trying distinct k aggregation functions. The second proposed algorithm improves on that. We repeat steps 1-5 of Algorithm 1. Once we have yAg1, . . . , yAgk Algorithm 2 is based on the calculation of all the possible combinations of yRAg1, yGAg1, yBAg1 (there are k^3 such combinations) in the following way:

\[
\begin{align*}
y_{\sigma_1} &= (yRAg_1, yGAg_1, yBAg_1) \\
y_{\sigma_2} &= (yRAg_1, yGAg_1, yBAg_2) \\
&\vdots \\
y_{\sigma_k} &= (yRAg_1, yGAg_k, yBAg_1) \\
y_{\sigma_{k+1}} &= (yRAg_2, yGAg_1, yBAg_1) \\
&\vdots \\
y_{\sigma_{k^3-1}} &= (yRAg_k, yGAg_k, yBAg_{k-1}) \\
y_{\sigma_{k^3}} &= (yRAg_k, yGAg_k, yBAg_k)
\end{align*}
\]

Notice that the possible outputs of Algorithm 1 are a subset of possible outputs of Algorithm 2. We analyze under which conditions solutions of Algorithm 2 are different from those in Algorithm 1. In these cases, the value of the same penalty function P with respect to y_{\sigma_1} will be less than the value y_{Ag_1} calculated in Algorithm 1. In Fig. 7 we show a diagram of Algorithm 2.

In Fig. 8 we apply Algorithm 2 in the same setting of Example 2. The first image (image (a)) is a synthetic one with

Algorithm 2 Second color image reduction algorithm

Input: Q of dimension N \times M 
Output: Q' of dimension N' \times M'
1: Divide the image Q in disjoint blocks of dimension n \times n.
   If N or M are not multiples of n eliminate the necessary number of rows and/or columns to satisfy this condition.
2: Choose a penalty function P.
3: Take k averaging aggregation functions Ag1, . . . , Agk.
4: for each block in Q do
5:   Apply to each pixel in each block (in the three channels R, G and B) k aggregation functions, as follows:
6:      yAg1 = (yRAg1, yGAg1, yBAg1) =
      \[ \begin{align*}
      \frac{Ag_1}{l=1...n} (x_{Rlj}), \frac{Ag_1}{j=1...n} (x_{Glj}), \frac{Ag_1}{i=1...n} (x_{Blj}) \\
      \ldots
      \end{align*} \]
7:      yAgk = (yRAgk, yGAgk, yBAgk) =
      \[ \begin{align*}
      \frac{Ag_k}{l=1...n} (x_{Rlj}), \frac{Ag_k}{j=1...n} (x_{Glj}), \frac{Ag_k}{i=1...n} (x_{Blj}) \\
      \ldots
      \end{align*} \]
8:      Calculate k^3 combinations of the values obtained in the previous step, as follows:
9:      y_{\sigma_i} = (y_{R\sigma_i(1)}, y_{G\sigma_i(2)}, y_{B\sigma_i(3)})
   where \sigma_i \in \{\sigma(1), \sigma(2), \sigma(3)\} \in \{1, 2, \ldots, k\}^3.
10: Calculate the penalties P_{\sigma_i} = P(x, y_{\sigma_i}) for each i = 1, 2, \ldots, k^3.
11: Assign the value y_{\sigma_i} with the smallest penalty to the corresponding pixel of the reduced image
12: \[ \arg\min_{y_{\sigma_i}} P(x, y_{\sigma_i}) \]
13: end for
small variations of color. The second image (image (b)) is a texture image with large variations of intensity. By analyzing the results we observe that when we apply Algorithms 1 and 2 to image (a) we obtain the same results. However, when we apply them to image (b), around 60% of pixels are different.

Hence when dealing with large intensity changes, the solutions given by Algorithm 2 provide smaller values of penalty $P$. However, the computational cost of this algorithm is higher. As the number of aggregations $\kappa$ increases, the running time of the Algorithm 2 increases exponentially, (whereas for Algorithm 1 it increases linearly). This prompted us to develop another algorithm improving on Algorithms 1 and 2.

C. Third algorithm for image reduction

The third reduction algorithm aims at identifying the global minimum of a penalty function $P$ for each block of the image. It is based on coordinate descent as outlined in Section IV. The idea of the algorithm is the following: first, we initialize the value of $y = (y_R, y_G, y_B)$. Then, for the first component, the goal of the coordinate descent is to find a value $a \in \mathbb{Z}$ such that $(y_R + a, y_G, y_B)$ is a minimum in $P$:

$$P(x, (y_R + a, y_G, y_B)) < P(x, (y_R + (a - 1), y_G, y_B))$$
and

$$P(x, (y_R + a, y_G, y_B)) < P(x, (y_R + (a + 1), y_G, y_B))$$

We apply the same process of coordinate descent to the second and the third components. Once we have the new value of $y$, we repeat the same process (minimization of the 3 components) until the value of $y$ remains the same for two consecutive iterations. Then $y$ is the value that minimizes the penalty function $P$. In Fig. 9 we show a diagram of Algorithm 3:

As was the case with Algorithm 2, the largest difference between images reduced by Algorithm 1 and Algorithm 3 can be found in the areas with bigger variation of intensities. In Fig. 10 we show, for each color component, an image of normalized differences between the images reduced with Algorithm 1 and the same images reduced with Algorithm 3. In these images, clearer pixels correspond to a bigger difference between the images. Observe that light areas correspond to edges, that is, areas with large changes of intensity.

VI. EXPERIMENTAL RESULTS

In this section we present a formal comparative study of the performance of the algorithms proposed in this work and some other image reduction algorithms from the literature. Other algorithms consider each color component separately, i.e. are based on the Cartesian product. We analyze 11 images in RGB of dimension $256 \times 256$ from the URL...
the obtained images to their respective original image without cost. Moreover, this method allows us to compare visually.

II-C. It is a very simple method with low computational complexity obviously depends on the dimension of the original

image. The method for enlargement is presented in Section II-C: to enlarge the reduced image to the original dimension and to compare the resulting image with the original image. We have changed 10% of the pixels in test images. We calculated MSE, SIM and PEN of the reconstructed images.

A. Experiments with impulsive noise

We now consider the same images with salt and pepper noise. We calculated MSE, SIM and PEN of the reconstructed images. We have changed 10% of the pixels in test images. We present the results in Tables VII, VIII and IX. In Fig. 14 we show the images obtained by means of Algorithm 1, fuzzy transform and subsampling.

In Table IV, V and VI we show the error in the reconstructions of the 11 images with noise levels of 5%, 10%, 20% and 30% (horizontal axis). Notice that the results of the three algorithms are very similar to each other and improve by around 18% the results of the fuzzy transform.

In Fig. 12 we visually show the results for two of the six images in Fig. 11 obtained by means of the three proposed reduction methods. To observe better the differences, in Fig. 13 we show the images reconstructed to its original size and obtained by means of Algorithm 1, fuzzy transform and subsampling.


In Fig. 11 we show the first 6 original images out of the 11 images considered.

We compare three proposed reduction algorithms with a classical subsampling algorithm (sub) (taking only one pixel from each block, usually the central one) and a recent method based on the fuzzy transform (Trans) [15]. We take as the penalty the function in Equation (2). To measure the accuracy of each method, we follow the same scheme presented in Section II-C: to enlarge the reduced image to the original dimension and to compare the resulting image with the original image. The method for enlargement is presented in Section II-C. It is a very simple method with low computational cost. Moreover, this method allows us to compare visually the obtained images to their respective original image without changing the results obtained for the reduced images.

To measure the differences between the reconstructed and the original images we use the error and similarity measures that appear most commonly in the literature (based on Cartesian products of similarities for each color), and a new measure based on the arithmetic mean in product lattices: the mean squared error (MSE), the similarity measure (SIM) presented in [25], [26] and the error based on the penalty (PEN) defined as

$$\text{MSE}(A,B) = \frac{\sum_{i=1}^{N} \sum_{j=1}^{M} (A_{ij} - B_{ij})^2}{3 \times N \times M}$$

$$\text{SIM}(A,B) = \frac{\sum_{i=1}^{N} \sum_{j=1}^{M} \left( \sum_{C \in \{R,G,B\}} \left| A_{ij} - B_{ij} \right| \right)}{3 \times N \times M}$$

$$\text{PEN}(A,B) = \frac{\sum_{i=1}^{N} \sum_{j=1}^{M} \left( \sum_{C \in \{R,G,B\}} |A_{ij} - B_{ij}| \right)^2}{3 \times N \times M}$$

In Fig. 12 we visually show the results for two of the six images in Fig. 11 obtained by means of the 5 analyzed reduction methods. To observe better the differences, in Fig. 13 we show the images reconstructed to its original size and obtained by means of Algorithm 1, fuzzy transform and subsampling.

A. Experiments with impulsive noise

We now consider the same images with salt and pepper noise. We calculated MSE, SIM and PEN of the reconstructed images. We have changed 10% of the pixels in test images. We present the results in Tables VII, VIII and IX. In Fig. 14 we show the images obtained by the 5 considered reduction algorithms (Alg 1, Alg 1, Alg 2, Alg 3, Trans and Sub) applied to the images of Fig. 11 with noise.

For images with the impulsive noise the three proposed algorithms provide the best results. In particular, results of Algorithm 1 are very competitive. In Section V-A2 we have already seen that, when adding salt and pepper noise, the number of times Algorithm 1 uses the median increases. In particular, we know that the median is very useful to suppress that kind of noise.

To analyze the algorithms confronting a larger amount of noise, in Fig. 16 we show the mean MSE, SIM and PEN respectively (vertical axis) of the reconstructions of the 11 images with noise levels of 5%, 10%, 20% and 30% (horizontal axis). Notice that the results of the three algorithms are very similar and much better when compared to the results obtained with the fuzzy transform and the subsampling methods, even when the amount of noise increases.

Finally, in Table X we present average running times of the three proposed algorithms and two methods we benchmark against. The algorithms have been programmed in Matlab. Obviously, the lowest runtime corresponds with subsampling algorithm because no computation on the image is required. However, the results of this algorithm are not good.

In the three presented algorithms, the computational complexity obviously depends on the dimension of the original

![Diagram of Algorithm 3](image3)

![Fig. 10. Differences in each color component between reductions obtained by Algorithm 1 and by Algorithm 3](image4)
image and on the size of the reduction block. But there are differences between the three algorithms. In Algorithm 1 and 2, the number of mathematical operations (calculation of the aggregation functions) is the same if we take the same value $k$ in step 3 of Algorithms 1 and 2. However, the number of evaluations of the penalty function is different: in Algorithm 1 we evaluate it $k$ times whereas in Algorithm 2 we evaluate it $k^3$ times. This increases the cost of the algorithm. Observe that the lowest runtime corresponds to Algorithm 1 while the run time of Algorithm 2 is very high. On the other side, in Algorithm 3, the number of evaluations of the penalty function depends on the coordinate descent algorithm. In noiseless images, the algorithm is able to find the minimum of the penalty function in two iterations. This makes the run time smaller than that of Algorithm 2. Moreover, the initialization value $y_0$ is determinant for the run time: An initialization close to the solution diminishes the required number of evaluations of $P$.

We now study CPU times of proposed algorithms for noisy images. We observe that CPU times of Algorithm 3 increase while the rest remain stable. If we add some extreme values (impulsive noise) to the data, then the number of iterations increases. Besides, Step 1 of coordinate descent algorithm (the initial value of $y_0$) is again important determinant of the number of iterations. In Table XI we show the CPU times of Algorithm 3 with three different initializations: 1) taking $y_0$ the cartesian product of arithmetic means ($y_0_{\text{Arith}}$); 2) taking $y_0$ the cartesian product of the medians ($y_0_{\text{Med}}$); 3) taking $y_0$ the result of applying Algorithm 1 ($y_0_{\text{Alg1}}$). Notice that the lowest CPU times corresponds to $y_0_{\text{Alg1}}$. As we have illustrated before, the results of Algorithm 1 are good solutions (Tables VII, VIII and IX) and appropriate initializations to the coordinate descent algorithm.

VII. CONCLUSIONS

Based on the representation of the classical mean and the median as solutions to minimization problems, we have defined the mean and the median on discrete product lattices in the same way. We have shown that the median becomes the Cartesian product of the medians defined on discrete chains,
Fig. 13. Reconstructed images of Fig. 12 of Alg 1, Trans and Sub reduction algorithms

Fig. 14. Reduced images of Fig. 11 with salt and pepper noise using algorithm 1, algorithm 2, algorithm 3, fuzzy transform and subsampling
Fig. 15. Reconstructed images of Fig. 12 of Alg 1, Trans and Sub reduction algorithms

Fig. 16. Mean MSE, SIM and PEN of 11 reconstructed images with 5%, 10%, 20% and 30% of salt and pepper noise density

and that the mean is not the Cartesian product of the respective means. We proved that penalty-based functions based on dissimilarities are monotone (i.e., aggregation functions) in the case of product lattices.

The main motivation and application of this work is aggregation of colors in image processing. In this context we have presented three image reduction algorithms based on aggregation by means of penalty functions. We have shown that, as in the case of gray scale images, the results obtained with different error measures may be determined by the aggregation function that is used.

We have compared the proposed algorithms with some of the most commonly used reduction methods, and we found that the proposed methods are superior for color image reduction. They are also robust with respect to impulsive noise in the images. We have studied the effect of noise on the proposed algorithms, and we found that they efficiently filter out non-Gaussian noise. The computational cost of the proposed methods is relatively small.

ACKNOWLEDGMENT

The authors would like to thank the editor and the anonymous reviewers for their valuable suggestions that have greatly improved the paper.

REFERENCES


### TABLE IV
MSE of reconstructions of 11 original images using algorithm 1, algorithm 2, algorithm 3, fuzzy transform and subsampling

<table>
<thead>
<tr>
<th>Images Alg 1</th>
<th>Alg 2</th>
<th>Alg 3</th>
<th>Trans</th>
<th>Sub</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alg 1</td>
<td>2</td>
<td>11</td>
<td>Alg 2</td>
<td>Alg 3</td>
</tr>
<tr>
<td>Alg 3</td>
<td>Alg 1</td>
<td>Alg 2</td>
<td>Alg 3</td>
<td>Alg 2</td>
</tr>
<tr>
<td>Alg 2</td>
<td>Alg 3</td>
<td>Alg 1</td>
<td>Alg 2</td>
<td>Alg 3</td>
</tr>
<tr>
<td>Alg 3</td>
<td>Alg 1</td>
<td>Alg 2</td>
<td>Alg 3</td>
<td>Alg 2</td>
</tr>
</tbody>
</table>

### TABLE V
SIM of reconstructions of 11 original images using algorithm 1, algorithm 2, algorithm 3, fuzzy transform and subsampling

<table>
<thead>
<tr>
<th>Images Alg 1</th>
<th>Alg 2</th>
<th>Alg 3</th>
<th>Trans</th>
<th>Sub</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alg 1</td>
<td>2</td>
<td>11</td>
<td>Alg 2</td>
<td>Alg 3</td>
</tr>
<tr>
<td>Alg 3</td>
<td>Alg 2</td>
<td>Alg 1</td>
<td>Alg 2</td>
<td>Alg 3</td>
</tr>
<tr>
<td>Alg 2</td>
<td>Alg 3</td>
<td>Alg 1</td>
<td>Alg 2</td>
<td>Alg 3</td>
</tr>
<tr>
<td>Alg 3</td>
<td>Alg 2</td>
<td>Alg 1</td>
<td>Alg 2</td>
<td>Alg 3</td>
</tr>
</tbody>
</table>

### TABLE VI
PEN of reconstructions of 11 original images using algorithm 1, algorithm 2, algorithm 3, fuzzy transform and subsampling

<table>
<thead>
<tr>
<th>Images Alg 1</th>
<th>Alg 2</th>
<th>Alg 3</th>
<th>Trans</th>
<th>Sub</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alg 1</td>
<td>2</td>
<td>11</td>
<td>Alg 2</td>
<td>Alg 3</td>
</tr>
<tr>
<td>Alg 3</td>
<td>Alg 2</td>
<td>Alg 1</td>
<td>Alg 2</td>
<td>Alg 3</td>
</tr>
<tr>
<td>Alg 2</td>
<td>Alg 3</td>
<td>Alg 1</td>
<td>Alg 2</td>
<td>Alg 3</td>
</tr>
<tr>
<td>Alg 3</td>
<td>Alg 2</td>
<td>Alg 1</td>
<td>Alg 2</td>
<td>Alg 3</td>
</tr>
</tbody>
</table>

### TABLE VII
Table VII

### TABLE VIII
SIM of reconstructions of 11 images with salt and pepper noise using algorithm 1, algorithm 2, algorithm 3, fuzzy transform and subsampling

<table>
<thead>
<tr>
<th>Images Alg 1</th>
<th>Alg 2</th>
<th>Alg 3</th>
<th>Trans</th>
<th>Sub</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alg 1</td>
<td>2</td>
<td>11</td>
<td>Alg 2</td>
<td>Alg 3</td>
</tr>
<tr>
<td>Alg 3</td>
<td>Alg 2</td>
<td>Alg 1</td>
<td>Alg 2</td>
<td>Alg 3</td>
</tr>
<tr>
<td>Alg 2</td>
<td>Alg 3</td>
<td>Alg 1</td>
<td>Alg 2</td>
<td>Alg 3</td>
</tr>
<tr>
<td>Alg 3</td>
<td>Alg 2</td>
<td>Alg 1</td>
<td>Alg 2</td>
<td>Alg 3</td>
</tr>
</tbody>
</table>

### TABLE IX
PEN of reconstructions of 11 images with salt and pepper noise using algorithm 1, algorithm 2, algorithm 3, fuzzy transform and subsampling

<table>
<thead>
<tr>
<th>Images Alg 1</th>
<th>Alg 2</th>
<th>Alg 3</th>
<th>Trans</th>
<th>Sub</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alg 1</td>
<td>2</td>
<td>11</td>
<td>Alg 2</td>
<td>Alg 3</td>
</tr>
<tr>
<td>Alg 3</td>
<td>Alg 2</td>
<td>Alg 1</td>
<td>Alg 2</td>
<td>Alg 3</td>
</tr>
<tr>
<td>Alg 2</td>
<td>Alg 3</td>
<td>Alg 1</td>
<td>Alg 2</td>
<td>Alg 3</td>
</tr>
<tr>
<td>Alg 3</td>
<td>Alg 2</td>
<td>Alg 1</td>
<td>Alg 2</td>
<td>Alg 3</td>
</tr>
</tbody>
</table>

### TABLE X
Mean CPU times of the reduction algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Med</th>
<th>1st</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alg 1</td>
<td>6.85</td>
<td>6.82</td>
<td>6.87</td>
</tr>
<tr>
<td>Alg 2</td>
<td>6.86</td>
<td>6.87</td>
<td>6.88</td>
</tr>
<tr>
<td>Alg 3</td>
<td>6.89</td>
<td>6.90</td>
<td>6.91</td>
</tr>
<tr>
<td>Avg</td>
<td>6.88</td>
<td>6.88</td>
<td>6.88</td>
</tr>
</tbody>
</table>

### TABLE XI
Mean CPU times of algorithm 3 with noisy images and different initializations.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Med</th>
<th>1st</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alg 1</td>
<td>6.85</td>
<td>6.82</td>
<td>6.87</td>
</tr>
<tr>
<td>Alg 2</td>
<td>6.86</td>
<td>6.87</td>
<td>6.88</td>
</tr>
<tr>
<td>Alg 3</td>
<td>6.89</td>
<td>6.90</td>
<td>6.91</td>
</tr>
<tr>
<td>Avg</td>
<td>6.88</td>
<td>6.88</td>
<td>6.88</td>
</tr>
</tbody>
</table>

Gleb Beliakov (SM’08) received the Ph.D. degree in physics and mathematics from the Russian Peoples Friendship University, Moscow, Russia, in 1992. He was a Lecturer and Research Fellow with Los Andes University, the Universities of Melbourne and South Australia. He is currently an Associate Professor with the School of IT, Deakin University. His research interests are in the areas of fuzzy systems, aggregation operators, multivariate approximation, global optimization, decision support systems, and applications of fuzzy systems in healthcare. He is the author of a hundred research papers in the mentioned areas and a number of software packages.

Daniel Paternain received the M.Sc. degree in Computer Sciences from the Public University of Navarra, Pamplona, Spain, in 2008. He is currently teaching assistant at the Department of Automatics and Computation where he is working towards the Ph.D. degree. His research interests are image processing, focusing on image reduction, and applications of aggregation functions, fuzzy sets and extensions of fuzzy sets.

Humberto Bustince is Full Professor at the Department of Automatics and Computation, Public University of Navarra, Spain. He holds a Ph.D. degree in Mathematics from Public University of Navarra from 1994. His research interests are fuzzy logic theory, extensions of Fuzzy sets (Type-2 fuzzy sets, Interval-valued fuzzy sets, Atanassov’s intuitionistic fuzzy sets), Fuzzy measures, Aggregation operators and fuzzy techniques for Image processing. Author of more than 80 papers in WoS.