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 products 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. We prove that, when we reconstruct a reduced image, the error with respect to the original image may be determined by the reduction method that has been employed.

We present three new color image reduction algorithms which are based on minimizing a penalty function defined over product lattices. We carry out an experimental study in which we compare the proposed algorithms with the alternative methods that can be found in the literature, and we analyze the stability of the algorithms with respect to noise in the images.

The structure of the paper is as follows. In Section II we provide preliminary definitions. In Section III we give the definitions of aggregation functions based on penalties, defined on product lattices and we present the problem of image reduction algorithms. We discuss solutions to resulting optimization problems in Section IV. In Section V we present the color image reduction algorithms and we present an experimental study in Section VI. Conclusions are presented in Section VII.

This work has been partially supported by research grant TIN2010-15055 from the Government of Spain.

G. Beliakov is with the School of Information Technology, Deakin University, Burwood 3125, Australia (e-mail: gleb@deakin.edu.au).

H. Bustince and D. Paternain are with the Departamento de Automatica y Computacion, Universidad Publica de Navarra, Pamplona 31006, Navarra, Spain (e-mail: bustince@unavarra.es,daniel.paternain@unavarra.es).

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 $f(\mathbf{a}) = a$, $f(\mathbf{b}) = b$, with $\mathbf{a} = (a, a, \ldots, a), \mathbf{b} = (b, b, \ldots, b)$.

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

$$\min(\mathbf{x}) := \min(x_1, \ldots, x_n) \le f(x_1, \ldots, x_n) \le$$
$$\le \max(x_1, \ldots, x_n) =: \max(\mathbf{x}).$$

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 $f(\mathbf{a}) = a$, $f(\mathbf{b}) = b$ 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 $\mathbf{x}$ 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 $P : [a, b]^{n+1} \to \Re$ be a penalty function with the properties

i) $P(\mathbf{x}, y) \geq 0$ for all $\mathbf{x}, y$;
ii) $P(\mathbf{x}, y) = 0$ if all $x_i = y$;
iii) $P(\mathbf{x}, y)$ is quasiconvex in y for any $\mathbf{x}$.

The penalty based function is

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

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 $f(ay_1 + (1 - a)y_2) \leq \max\{f(y_1), f(y_2)\}$ 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

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

whereas the median is a solution to

$$minimize_y \sum_{i=1}^{n} |x_i - y|.$$

In this work we will deal with penalty based functions defined on discrete lattices, rather than the interval $[a, b]$.

B. Lattices

Definition 4: Let L be a set. A lattice $\mathcal{L} = (L, \leq, \wedge, \vee)$ is a poset with the partial order $\leq$ on L, and meet and join operations $\wedge, \vee$, if every pair of elements from L has both meet and join.

Definition 5: Let P be a poset. A chain in P is a totally ordered subset of P. The length of a chain is its cardinality.

Definition 6: If $\mathcal{L}_1 = (L_1, \leq_1, \wedge_1, \vee_1)$ and $\mathcal{L}_2 = (L_2, \leq_2, \wedge_2, \vee_2)$ are two lattices, their Cartesian product is the lattice $\mathcal{L}_1 \times \mathcal{L}_2 = (L_1 \times L_2, \leq, \wedge, \vee)$ with $\leq$ defined by

$$(x_1, y_1) \leq (x_2, y_2) \Leftrightarrow x_1 \leq_1 x_2 \text{ and } y_1 \leq_2 y_2,$$

and

$$(x_1, y_1) \wedge (x_2, y_2) = (x_1 \wedge_1 x_2, y_1 \wedge_2 y_2),$$

$$(x_1, y_1) \vee (x_2, y_2) = (x_1 \vee_1 x_2, y_1 \vee_2 y_2).$$

We will deal with Cartesian products of finite chains $\mathcal{C}$, which is precisely the type of product lattice representing colors in image processing, with the length of each chain typically being 256. We note that all finite chains of the same length are isomorph to each other, and hence we can represent them as non-negative integers $0, 1, \ldots, K$, and elements of product lattices as tuples $x = (x_1, x_2 \ldots, x_m)$, $x_i \in Z_+ = \{0, 1, 2, \ldots\}$.

Definition 7: Let f_1, f_2 be two aggregation functions defined on sets X_1 and X_2 respectively. The Cartesian product of aggregation functions is $f = f_1 \times f_2 : X_1 \times X_2 \to Y_1 \times Y_2$ defined by

$$f(\mathbf{x}_1, \mathbf{x}_2) = (f_1(\mathbf{x}_1), f_2(\mathbf{x}_2)).$$

C. Image reduction

Image reduction consists in reducing the dimension of the image while keeping as much information as possible. Image reduction can be used to accelerate computations on an image, or just to reduce the cost of its storage or transmission.

There exist several methods for image reduction in the literature. Some of them consider the image to be reduced in a global way [15]–[17] or in a transform domain [18]. Other widely used methods act locally over pieces (blocks) of the image [19], [20]. The division of the image in blocks of small size allows one to design simple reduction algorithms.

In this work, we consider an image of $N \times M$ pixels as a set of $N \times M$ elements arranged in rows and columns. Each element of a grayscale image is represented by x_{ij} with $i \in \{1, \ldots, N\}$ and $j \in \{1, \ldots, M\}$. The element x_{ij} has a value between 0 and $L - 1$. If we consider color image in the RGB reference system, each element of the image is denoted by $x_{ij} = (x_{Rij}, x_{Gij}, x_{Bij})$. Each color component will also have a value between 0 and $L - 1$.

A typical local image reduction algorithm is presented as follows:

Input: Q of dimension $N \times M$
Output: Q' of dimension $\frac{N}{n} \times \frac{M}{n}$

1: Divide the image Q into disjoint blocks of dimension $n \times n$. If N or M are not multiples of n eliminate the smallest number of rows and/or columns to satisfy this condition.
2: Choose an averaging function f.
3: **for** each block in Q **do**
4: Calculate $f(x_{11}, \ldots, x_{1n}, \ldots, x_{nn})$.
5: Place the result in the corresponding pixel of the reduced image (see Fig. 1).
6: **end for**

In Fig. 2 we show three reduced images obtained from the original image Lena using the following aggregation functions in step 2 of previous algorithm: the geometric mean (b), the arithmetic mean (c) and the median (d).

There exist a number of methods to determine which is the best reduction. One of the most frequently used methods is the following:

1) Magnify the reduced image to the dimensions of the original image.
2) Measure the error between the reconstructed and the original image.

There exist different image magnification methods that will influence the final result [21], [22]. However, in this work we

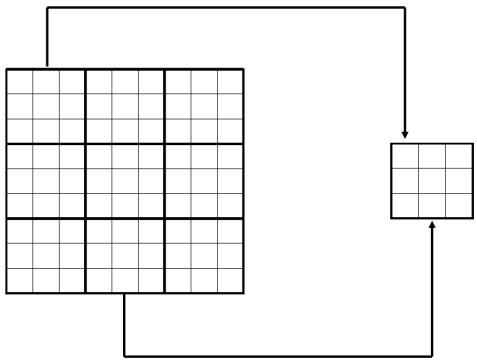


Fig. 1. Scheme of the reduction algorithm

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

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.

	Image (b1)	Image (b2)	Image (b3)
MSE	224.0065	**209.23**	237.44
MAE	7.89	7.88	**7.42**

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

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 $\mathcal{L} = (L, \leq, \wedge, \vee)$ be a product of finite chains. The distance between $x, y \in \mathcal{L}$ is defined as the length of a maximal chain $\mathcal{C}$ with the least element $a = x \wedge y$ and the greatest element $b = x \vee y$ minus 1,

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

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

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

Remark 3: P is quasi-convex in y, as in Definition 3. But now $y \in \mathcal{L}$ 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 $f : X \subseteq \Re^n \to R$ is quasi-convex on X if all its level sets are convex. We call the function $f : \mathcal{L} \subseteq Z_+^n \to \Re$ quasi-convex if its extension $\bar{f} : X \to \Re$ is quasi-convex, where $X \subseteq \Re^n$ is the smallest convex set containing $\mathcal{L}$. Similarly $f : \mathcal{L} \to \Re$ is convex if its extension $\bar{f}$ is convex.

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

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

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

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

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

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 $\mathbf{x}$ and $\mathbf{x}'$, such that $x'_{ij} = x_{ij}$ for all i, j except one pair, $x'_{kl} = x_{kl} + 1$. We need to show that if $P(\mathbf{x}, y^*) \leq P(\mathbf{x}, y)$ for all $y \in \mathcal{L}$, then $P(\mathbf{x}', y^*) < P(\mathbf{x}', y)$ for all $y < y^*$.

Suppose $y < y^*$. Take $u = \sum_{j=1}^{m} = \sum_{j \neq l} |x_{kj} - y^*_j| + |x_{kl} + 1 - y^*_l|$, $v = \sum_{j=1}^{m} |x_{kj} - y^*_j|$, $u' = \sum_{j \neq l} |x_{kj} - y_j| + |x_{kl} + 1 - y_l|$, $v' = \sum_{j=1}^{m} |x_{kj} - y_j|$. Now, $u - v = |x_{kl} + 1 - y^*_l| - |x_{kl} - y^*_l|$, which is $+1$ if $y^*_l \leq x_{kl}$ or -1 otherwise. Also $u' - v' = |x_{kl} + 1 - y_l| - |x_{kl} - y_l|$, which is also either $+1$ or -1, but because $y < y^*$, it takes value -1 only if $u - v = -1$. Hence $u - v \leq u' - v'$.

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

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

■

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

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

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

$$minimize_y \sum_{i=1}^{n} \sum_{j=1}^{m} |x_{ij} - y_j| = \sum_{j=1}^{m} \sum_{i=1}^{n} |x_{ij} - y_j|. \tag{3}$$

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 $y = (Med(x_{\cdot,1}), \ldots, Med(x_{\cdot,m}))$, 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 $\frac{N}{n} \times \frac{N}{n}$

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 $Ag_1, \ldots, Ag_k$.
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:

$$y_{Ag_1} = (y_{RAg_1}, y_{GAg_1}, y_{BAg_1}) =$$
$$= \left(\underset{\substack{i=1..n \\ j=1..n}}{\mathrm{Ag}_1}\ (x_{Rij}),\ \underset{\substack{i=1..n \\ j=1..n}}{\mathrm{Ag}_1}\ (x_{Gij}),\ \underset{\substack{i=1..n \\ j=1..n}}{\mathrm{Ag}_1}\ (x_{Bij}) \right)$$
$$\ldots$$
$$y_{Ag_k} = (y_{RAg_k}, y_{GAg_k}, y_{BAg_k}) =$$
$$= \left(\underset{\substack{i=1..n \\ j=1..n}}{\mathrm{Ag}_k}\ (x_{Rij}),\ \underset{\substack{i=1..n \\ j=1..n}}{\mathrm{Ag}_k}\ (x_{Gij}),\ \underset{\substack{i=1..n \\ j=1..n}}{\mathrm{Ag}_k}\ (x_{Bij}) \right)$$

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

$$\arg\min_{y_{Ag_i}} P(\mathbf{x}, y_{Ag_i})$$

8: **end for**

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

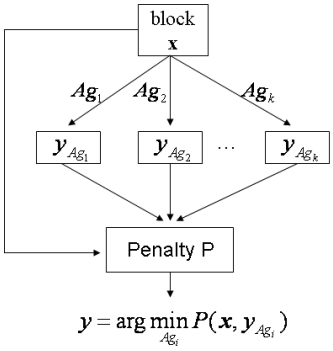


Fig. 3. Diagram of Algorithm 1

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

$\mathbf{x}_R$			$\mathbf{x}_G$			$\mathbf{x}_B$		
10	15	15	115	100	120	221	220	220
12	12	13	130	125	125	150	200	210
10	11	11	135	130	130	215	200	215

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

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

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

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

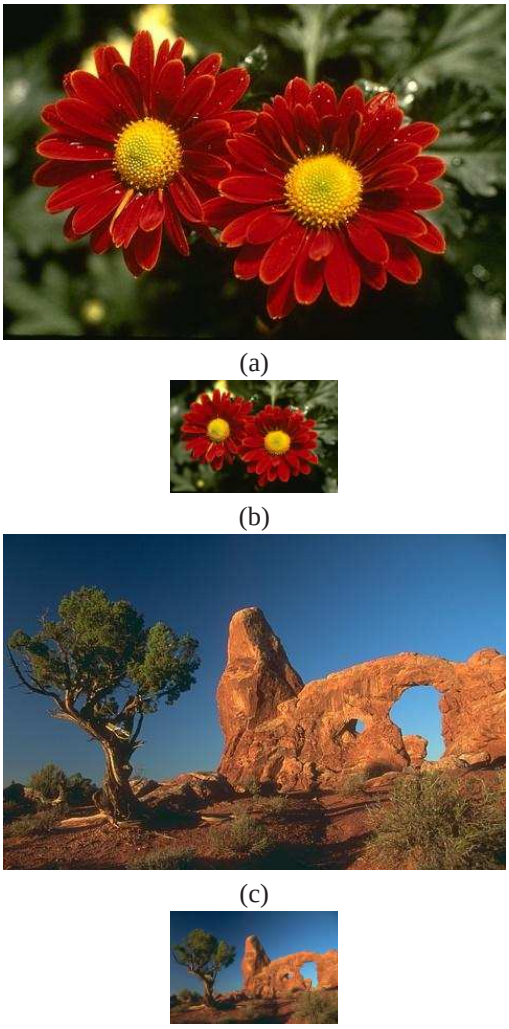
For this block of the image we take y_{med}, which corresponds to the smallest value of the penalty P for this block. Note that even though the penalty function corresponds to the arithmetic mean, the solution is not y_{arith} (which corresponds to the Cartesian product of arithmetic means), which is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

	Min	*Geom*	*Arith*	*Med*	*Max*	*Any*
Im (a)	0.03%	6.14%	63.08%	26.69%	0.02%	4.04%
Im (c)	0.03%	5.43%	70.50%	17.72%	0.05%	6.27%

TABLE II
Frequency of choosing aggregation functions by Algorithm 1 in images (a) and (c) of Fig. 4



(a) (b) (c) (d)

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

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 firtst 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.

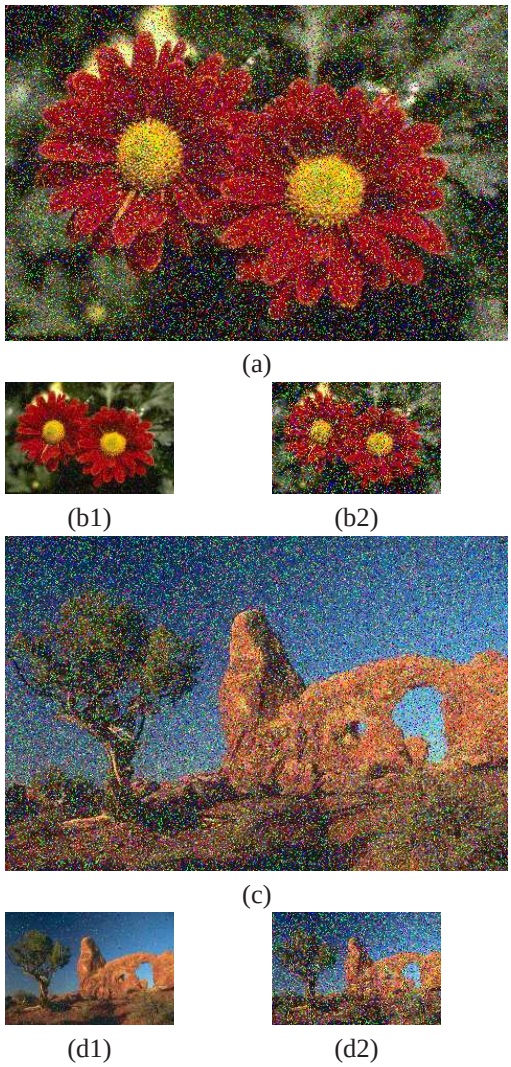
Notice that when the amount of salt and pepper noise in the images increases, the frequency of choosing the median also increases. This can be seen in Fig. 6. On the horizontal axis we show the percentage of pixels affected by noise. On the vertical axis we show the percentage of times that each aggregation 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.

Image and noise density	*Min*	*Geom*	*Arith*	*Med*	*Max*
Im (a) 5%	0.15%	6.90%	62.76%	30.13%	0.06%
Im (c) 5%	0.14%	8.25%	61.09%	30.42%	0.10%
Im (a) 10%	0.20%	5.44%	51.17%	43.12%	0.07%
Im (c) 10%	0.15%	5.76%	46.13%	47.86%	0.10%
Im (a) 20%	0.14%	2.60%	34.87%	62.37%	0.02%
Im (c) 20%	0.14%	1.97%	28.41%	69.42%	0.06%
Im (a) 30%	0.09%	1.55%	27.55%	70.81%	0.00%
Im (c) 30%	0.06%	0.73%	24.99%	74.19%	0.03%

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

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.



(a) (b1) (b2) (c) (d1) (d2)

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).

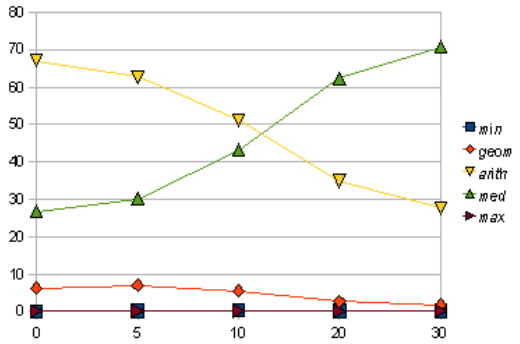


Image (a)

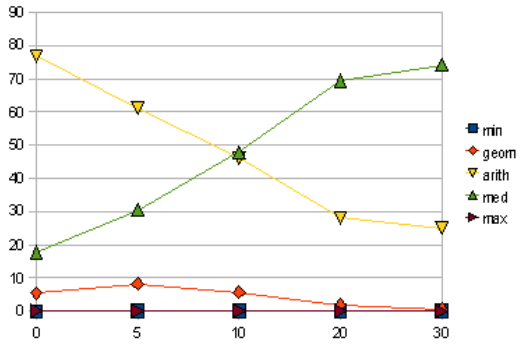


Image (c)

Fig. 6. Frequency of aggregation functions as a function of the intensity of the salt and pepper noise of original images (a) and (c) of Fig. 4

B. 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 $y_{Ag_1}, \ldots, y_{Ag_k}$ Algorithm 2 is based on the calculation of all the possible combinations of $y_{RAg_i}, y_{GAg_j}, y_{BAg_l}$ (there are k^3 such combinations) in the following way:

$$\begin{aligned}
y_{\sigma_1} &= (y_{RAg_1}, y_{GAg_1}, y_{BAg_1}) \\
y_{\sigma_2} &= (y_{RAg_1}, y_{GAg_1}, y_{BAg_2}) \\
&\ldots \\
y_{\sigma_k} &= (y_{RAg_1}, y_{GAg_1}, y_{BAg_k}) \\
y_{\sigma_{k+1}} &= (y_{RAg_2}, y_{GAg_1}, y_{BAg_1}) \\
&\ldots \\
y_{\sigma_{k^3-1}} &= (y_{RAg_k}, y_{GAg_k}, y_{BAg_{k-1}}) \\
y_{\sigma_{k^3}} &= (y_{RAg_k}, y_{GAg_k}, y_{BAg_k})
\end{aligned}$$

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_{σ_i} will be less than the value y_{Ag_j} 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 $\frac{N}{n} \times \frac{N}{n}$

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 $Ag_1, \ldots, Ag_k$.

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:

$$y_{Ag_1} = (y_{RAg_1}, y_{GAg_1}, y_{BAg_1}) =$$

$$= \left(\underset{\substack{i=1..n \\ j=1..n}}{\mathrm{Ag}_1}\ (x_{Rij}),\ \underset{\substack{i=1..n \\ j=1..n}}{\mathrm{Ag}_1}\ (x_{Gij}),\ \underset{\substack{i=1..n \\ j=1..n}}{\mathrm{Ag}_1}\ (x_{Bij}) \right)$$

$$\ldots$$

$$y_{Ag_k} = (y_{RAg_k}, y_{GAg_k}, y_{BAg_k}) =$$

$$= \left(\underset{\substack{i=1..n \\ j=1..n}}{\mathrm{Ag}_k}\ (x_{Rij}),\ \underset{\substack{i=1..n \\ j=1..n}}{\mathrm{Ag}_k}\ (x_{Gij}),\ \underset{\substack{i=1..n \\ j=1..n}}{\mathrm{Ag}_k}\ (x_{Bij}) \right)$$

6: Calculate k^3 combinations of the values obtained in the previous step, as follows:

$$y_{\sigma_i} = (y_{R\sigma(1)}, y_{G\sigma(2)}, y_{B\sigma(3)})$$

where $\sigma_i = (\sigma(1), \sigma(2), \sigma(3)) \in \{1, 2, \ldots, k\}^3$.

7: Calculate the penalties $P_{\sigma_i} = P(\mathbf{x}, y_{\sigma_i})$ for each $i = 1, \ldots, k^3$.

8: Assign the value y_{σ_i} with the smallest penalty to the corresponding pixel of the reduced image

$$\arg\min_{y_{\sigma_i}} P(\mathbf{x}, y_{\sigma_i})$$

9: **end for**

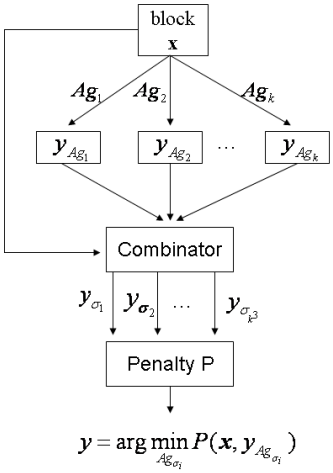


Fig. 7. Diagram of Algorithm 2

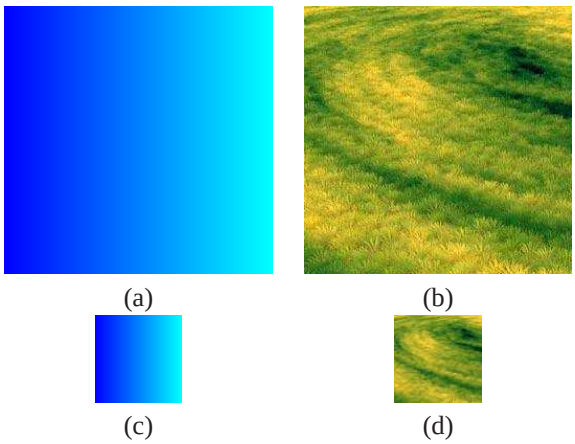


Fig. 8. Original images (a) and (b) and color image reductions (c) and (d) obtained by Algorithm 2

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 k 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 $y = (y_{R0}, y_{G0}, y_{B0})$. 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:

$$P(\mathbf{x}, (y_R + a, y_G, y_b)) < P(\mathbf{x}, (y_R + (a-1), y_G, y_B))$$

and

$$P(\mathbf{x}, (y_R + a, y_G, y_b)) < P(\mathbf{x}, (y_R + (a+1), y_G, y_B))$$

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.

Algorithm 3 Third color image reduction algorithm

Input: Q of dimension $N \times M$
Output: Q' of dimension $\frac{N}{n} \times \frac{N}{n}$

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 the penalty function P.
3: **for** each block in Q **do**
4: Calculate $\mathbf{y}* = \arg\min_{\mathbf{y}} P(\mathbf{x}, \mathbf{y})$ by means of the coordinate descent algorithm (Algorithm 4)
5: Assign the value $\mathbf{y}*$ to the corresponding pixel of the reduced image.
6: **end for**

Algorithm 4 Coordinate descent Algorithm

Input: $y_0 = (y_{R0}, y_{G0}, y_{B0})$
Output: y

1: $y \leftarrow (y_{R0}, y_{G0}, y_{B0})$
2: **repeat**
3: **for** each component $\{R, G, B\}$ **do**
4: **if** component=R **then**
5: $y_1 = (a, 0, 0);\ y_2 = (a+1, 0, 0)$
6: **else if** component=G **then**
7: $y_1 = (0, a, 0);\ y_2 = (0, a+1, 0)$
8: **else**
9: $y_1 = (0, 0, a);\ y_2 = (0, 0, a+1)$
10: **end if**
11: $a \leftarrow 1$
12: **if** $P(\mathbf{x}, y + y_1) < P(\mathbf{x}, y)$ **then**
13: **repeat**
14: $a \leftarrow a + 1$
15: update y_1, y_2
16: **until** $P(\mathbf{x}, y + y_1) < P(\mathbf{x}, y + y_2)$
17: $y \leftarrow y + y_1$
18: **else**
19: **if** $P(\mathbf{x}, y - y_1) < P(\mathbf{x}, y)$ **then**
20: **repeat**
21: $a \leftarrow a + 1$
22: update y_1, y_2
23: **until** $P(\mathbf{x}, y - y_1) < P(\mathbf{x}, y - y_2)$
24: $y \leftarrow y - y_1$
25: **end if**
26: **end if**
27: **end for**
28: **until** y is no longer modified

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 256×256 from the URL

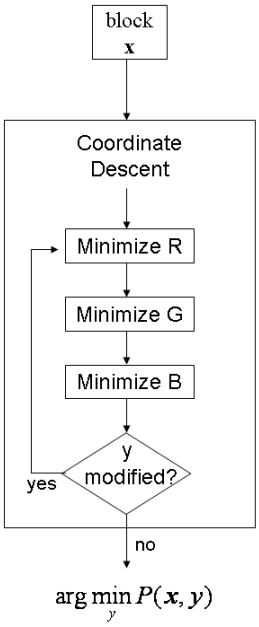


Fig. 9. Diagram of Algorithm 3

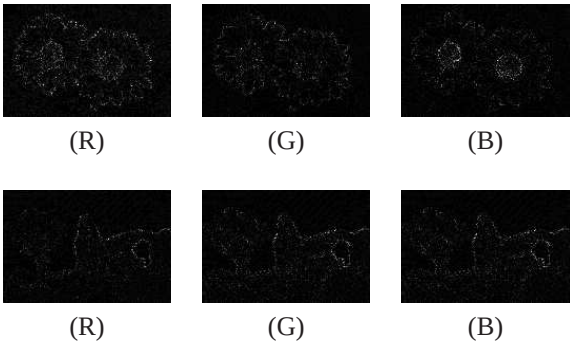


Fig. 10. Differences in each color component between reductions obtained by Algorithm 1 and by Algorithm 3

`http://decsai.ugr.es/cvg/dbimagenes/index.php`. 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

$$MSE(A,B) = \frac{\sum_{C\in\{R,G,B\}}\sum_{i=1}^{N}\sum_{j=1}^{M}(A_{Cij}-B_{Cij})^2}{3\times N\times M}$$

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

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

In Table IV, V and VI we show the error in the reconstruction by using MSE, SIM and PEN respectively. The smaller values of MSE and PEN, the better, and the larger SIM, the better. The best results are obtained with the algorithms that we propose in this work. Moreover, 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 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

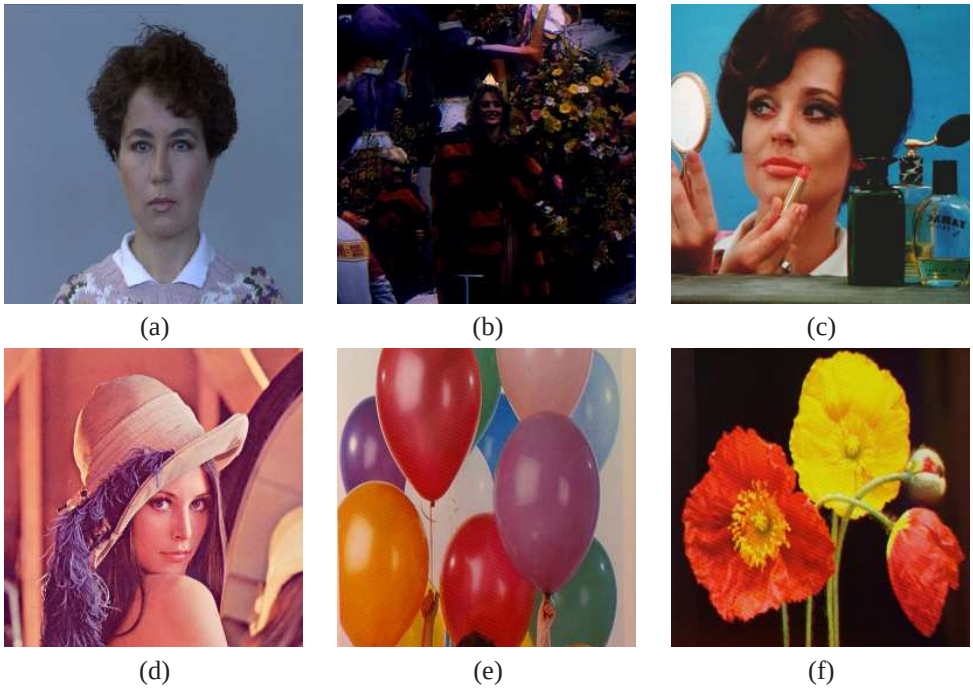


Fig. 11. Original images used in the experimental study

Fig. 12. Reduced images of Im (d) and (e) of Fig. 11 using algorithm 1, algorithm 2, algorithm 3, fuzzy transform and subsampling

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_{0Arith}); 2) taking y_0 the cartesian product of the medians (y_{0Med}); 3) taking y_0 the result of applying Algorithm 1 (y_{0Alg1}). Notice that the lowest CPU times corresponds to y_{0Alg1}. 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

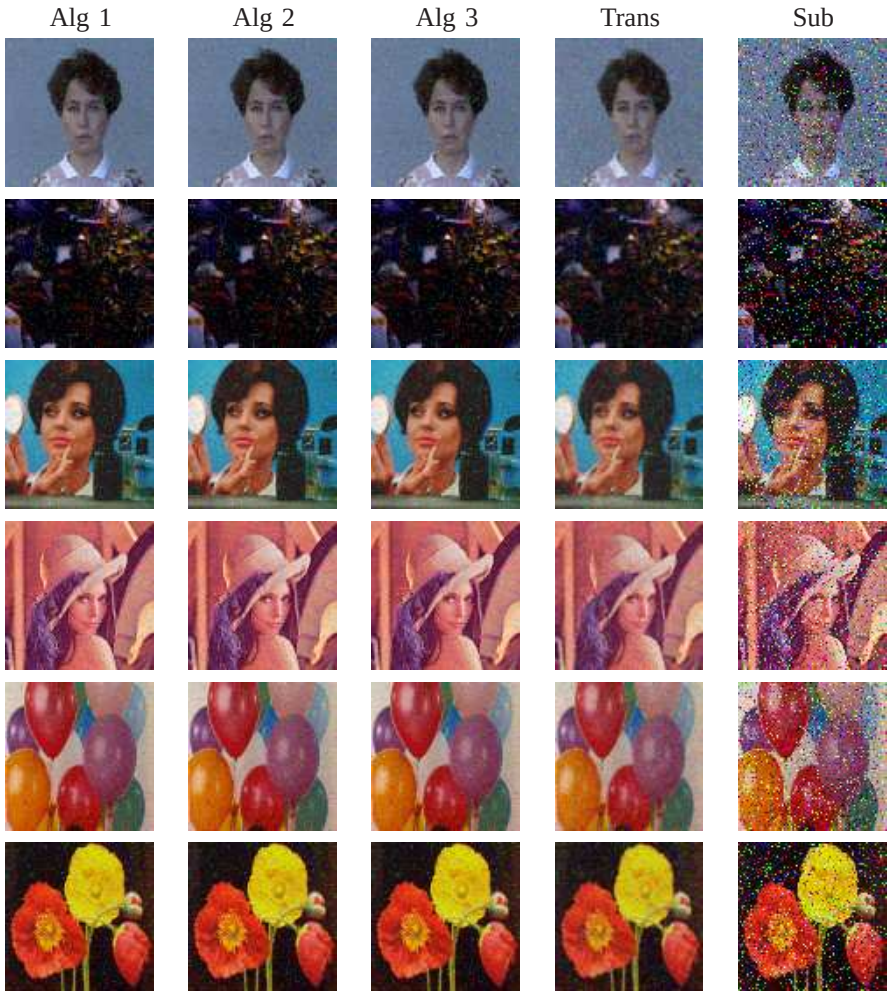


Fig. 14. Reduced images of Fig. 11 with salt and pepper noise using algorithm 1, algorithm 2, algorithm 3, fuzzy transform and subsampling

Fig. 15. Reconstructed images of Fig. 12 of Alg 1, Trans and Sub reduction algorithms

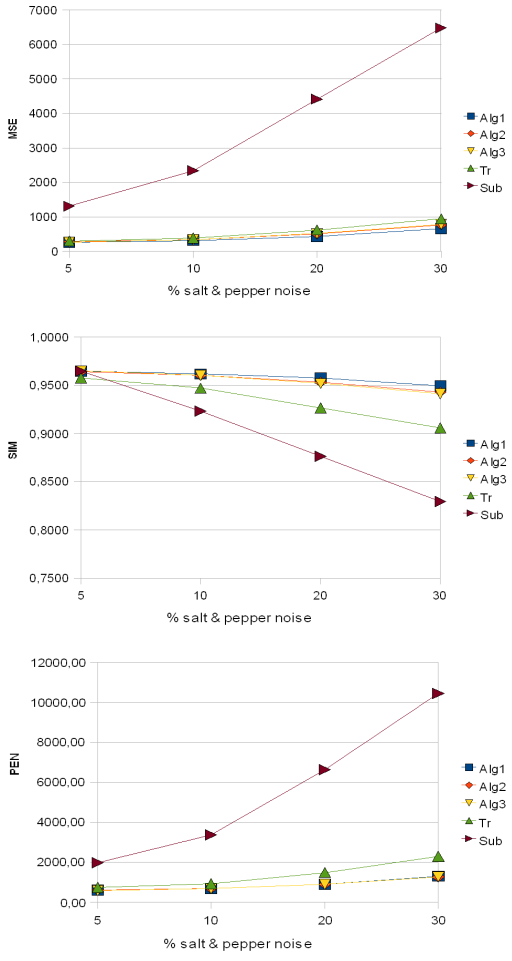


Fig. 16. Mean MSE, SIM and PEN of 11 reconstructed images with 5%, 10%, 20% and 30% of salt and pepper noise density

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

[1] M. Galar, J. Fernandez, G. Beliakov, and H. Bustince, “Interval-valued fuzzy sets applied to stereo matching of color images,” *IEEE Transactions on Image Processing*, vol. 20, pp. 1949–1961, 2011.

[2] B. De Baets and R. Mesiar, “Triangular norms on product lattices,” *Fuzzy Sets and Systems*, vol. 104, pp. 61–75, 1999.

[3] S. Jenei and B. De Baets, “On the direct decomposability of t-norms on product lattices,” *Fuzzy Sets and Systems*, vol. 139, pp. 699–707, 2003.

[4] G. Mayor and J. Monreal, “Additive generators of discrete conjunctive aggregation operations,” *IEEE Trans. on Fuzzy Systems*, vol. 15, pp. 1046–1052, 2007.

Images	Alg 1	Alg 2	Alg 3	Trans	Sub
2	**45.36**	45.43	45.52	55.79	63.82
4	**167.63**	169.03	170.05	204.22	235.59
5	**125.09**	125.23	125.97	156.83	162.99
6	**192.30**	193.08	193.92	232.22	280.28
7	**78.42**	78.59	78.94	100.20	104.36
8	**123.28**	123.82	124.13	154.08	153.41
1	**242.21**	242.48	242.82	283.10	357.46
3	**46.59**	46.67	46.77	56.94	65.26
9	**291.69**	293.46	293.23	363.96	414.39
10	**418.21**	419.50	421.13	518.82	583.53
11	**344.22**	345.36	346.00	396.89	572.09
Avg	**188.63**	189.33	189.83	229.46	272.11

TABLE IV
MSE OF RECONSTRUCTIONS OF 11 ORIGINAL IMAGES USING ALGORITHM 1, ALGORITHM 2, ALGORITHM 3, FUZZY TRANSFORM AND SUBSAMPLING

Images	Alg 1	Alg 2	Alg 3	Trans	Sub
2	0.9881	**0.9881**	0.9881	0.9867	0.9867
4	0.9758	0.9759	**0.9759**	0.9717	0.9737
5	0.9792	0.9792	**0.9792**	0.9755	0.9780
6	0.9692	**0.9693**	0.9693	0.9648	0.9654
7	0.9823	**0.9823**	0.9823	0.9648	0.9654
8	0.9788	0.9788	**0.9789**	0.9796	0.9805
1	0.9641	0.9641	**0.9642**	0.9600	0.9610
3	0.9879	**0.9879**	0.9879	0.9866	0.9865
9	0.9668	0.9668	**0.9669**	0.9609	0.9656
10	0.9542	0.9543	**0.9544**	0.9460	0.9520
11	0.9551	0.9552	**0.9552**	0.9510	0.9470
Avg	0.9729	0.9729	**0.9729**	0.9690	0.9703

TABLE V
SIM OF RECONSTRUCTIONS OF 11 ORIGINAL IMAGES USING ALGORITHM 1, ALGORITHM 2, ALGORITHM 3, FUZZY TRANSFORM AND SUBSAMPLING

Images	Alg 1	Alg 2	Alg 3	Trans	Sub
2	128.06	127.93	**127.78**	158.14	181.30
4	406.82	404.01	**401.60**	514.78	574.39
5	299.04	297.66	**296.54**	383.32	392.73
6	513.53	512.39	**511.12**	630.04	749.47
7	191.65	190.91	**190.20**	250.07	263.74
8	278.04	276.68	**275.42**	351.30	358.69
1	703.92	703.51	**703.03**	824.69	1042.1
3	131.70	131.57	**131.41**	161.34	185.44
9	775.64	774.34	**772.98**	965.47	1123.8
10	1048.1	1044.5	**1041.3**	1312.0	1507.0
11	970.28	968.67	**967.22**	1123.9	1631.0
Avg	495.16	493.83	**492.60**	606.82	728.15

TABLE VI
PEN OF RECONSTRUCTIONS OF 11 ORIGINAL IMAGES USING ALGORITHM 1, ALGORITHM 2, ALGORITHM 3, FUZZY TRANSFORM AND SUBSAMPLING

Images	Alg 1	Alg 2	Alg 3	Trans	Sub
2	**127.30**	143.86	141.75	167.51	1835.4
4	**390.85**	417.05	410.49	467.00	3211.5
5	**252.41**	272.94	269.27	316.59	2274.9
6	**303.60**	327.37	322.50	369.40	2264.7
7	**181.28**	201.62	199.76	235.58	2093.3
8	**297.71**	316.10	306.02	376.42	2698.4
1	**355.60**	376.94	384.69	404.07	2237.4
3	**127.85**	144.98	146.51	167.78	1970.2
9	**423.57**	576.17	468.89	509.31	2415.6
10	**565.27**	585.01	601.84	673.45	2640.2
11	**453.35**	473.18	492.14	510.35	2385.5
Avg	**316.25**	348.66	340.35	381.59	2366.1

TABLE VII
MSE OF RECONSTRUCTIONS OF 11 IMAGES WITH SALT AND PEPPER NOISE USING ALGORITHM 1, ALGORITHM 2, ALGORITHM 3, FUZZY TRANSFORM AND SUBSAMPLING

Images	Alg 1	Alg 2	Alg 3	Trans	Sub
2	**0.9785**	0.9760	0.9761	0.9638	0.9386
4	**0.9575**	0.9565	0.9566	0.9379	0.9268
5	**0.9670**	0.9653	0.9653	0.9510	0.9309
6	**0.9603**	0.9586	0.9589	0.9469	0.9185
7	**0.9713**	0.9586	0.9589	0.9570	0.9326
8	**0.9627**	0.9616	0.9625	0.9466	0.9313
1	**0.9550**	0.9539	0.9535	0.9431	0.9124
3	**0.9782**	0.9756	0.9756	0.9636	0.9372
9	**0.9565**	0.9547	0.9552	0.9413	0.9185
10	**0.9451**	0.9444	0.9442	0.9284	0.9067
11	**0.9477**	0.9467	0.9464	0.9384	0.9028
Avg	**0.9618**	0.9602	0.9604	0.9471	0.9233

TABLE VIII
SIM OF RECONSTRUCTIONS OF 11 IMAGES WITH SALT AND PEPPER NOISE USING ALGORITHM 1, ALGORITHM 2, ALGORITHM 3, FUZZY TRANSFORM AND SUBSAMPLING

Images	Alg 1	Alg 2	Alg 3	Trans	Sub
2	**246,44**	252,98	249,91	395,96	2433,2
4	736,75	722,86	**712,55**	1091,1	3917,6
5	493,17	**479,72**	480,04	726,44	3021,5
6	685,18	687,80	**674,40**	902,41	3378,4
7	363,52	363,38	**352,46**	545,31	2758,4
8	540,33	532,08	**512,94**	796,97	3277,2
1	873,03	863,29	**862,20**	1077,5	3678,9
3	**252,07**	260,92	259,93	402,80	2559,4
9	972,20	972,28	**956,85**	1267,8	3819,3
10	1272,4	1249,2	**1244,9**	1673,4	4220,7
11	1136,3	1130,5	**1129,7**	1332,8	4117,3
Avg	688.31	683.18	**675.99**	928.41	3380.2

TABLE IX
PEN OF RECONSTRUCTIONS OF 11 IMAGES WITH SALT AND PEPPER NOISE USING ALGORITHM 1, ALGORITHM 2, ALGORITHM 3, FUZZY TRANSFORM AND SUBSAMPLING

[5] T. Calvo and G. Beliakov, "Aggregation functions based on penalties," *Fuzzy Sets and Systems*, vol. 161, pp. 1420–1436, 2010.

[6] T. Calvo, R. Mesiar, and R. Yager, "Quantitative weights and aggregation," *IEEE Trans. on Fuzzy Systems*, vol. 12, pp. 62–69, 2004.

[7] R. Yager and A. Rybalov, "Understanding the median as a fusion operator," *Int. J. General Syst.*, vol. 26, pp. 239–263, 1997.

[8] R. Mesiar, "Fuzzy set approach to the utility, preference relations, and aggregation operators," *Europ. J. Oper. Res.*, vol. 176, pp. 414–422, 2007.

[9] G. Beliakov, A. Pradera, and T. Calvo, *Aggregation Functions: A Guide for Practitioners*. Heidelberg, Berlin, New York: Springer, 2007.

[10] Y. Torra, V. Narukawa, *Modeling Decisions. Information Fusion and Aggregation Operators*. Berlin, Heidelberg: Springer, 2007.

[11] T. Calvo, G. Mayor, and R. Mesiar, Eds., *Aggregation Operators. New Trends and Applications*. Heidelberg, New York: Physica-Verlag, 2002.

[12] M. Grabisch, J.-L. Marichal, R. Mesiar, and E. Pap, *Aggregation Functions*. Cambridge: Cambridge University press, 2009.

[13] H. Bustince, T. Calvo, B. De Baets, J. Fodor, R. Mesiar, J. Montero, D. Paternain, and A. Pradera, "A class of aggregations functions en-

	Alg 1	Alg 2	Alg 3	Trans	Sub
Mean runtime	1.65s	27.4s	5.86s	36.2s	4.4×10^{-5}s

TABLE X
MEAN CPU TIMES OF THE REDUCTION ALGORITHMS

	$\mathbf{y}_{0Arith}$	$\mathbf{y}_{0Med}$	$\mathbf{y}_{0Alg1}$
Mean runtime	16.94s	24.61s	14.08s

TABLE XI
MEAN CPU TIMES OF ALGORITHM 3 WITH NOISY IMAGES AND DIFFERENT INITIALIZATIONS.

compassing two-dimensional owa operators," *Information Sciences*, vol. 180, pp. 1977–1989, 2010.
[14] C. Gini, *Le Medie*. Milan (Russian translation, Srednie Velichiny, Statistica, Moscow, 1970): Unione Tipografico-Editorial Torinese, 1958.
[15] I. Perfilieva, "Fuzzy transforms: Theory and applications," *Fuzzy Sets and Systems*, vol. 157, pp. 993–1023, 2006.
[16] H. Nobuhara, K. Hirota, S. Sessa, and W. Pedrycz, "Efficient decomposition methods of fuzzy relation and their application to image decomposition," *Applied Soft Computing*, vol. 5, pp. 399–408, 2005.
[17] H. Kirshner and M. Porat, "On the role of exponential splines in image interpolation," *IEEE Transactions on Image Processing*, vol. 18, pp. 2198–2208, 2009.
[18] Y. Park and H. Park, "Arbitrary-ratio image resizing using fast dct of composite length for dct based transcode," *IEEE Transactions on Image Processing*, vol. 15, pp. 494–500, 2006.
[19] F. Di Martino, V. Loia, I. Perfilieva, and S. Sessa, "An image coding/decoding method based on direct and inverse fuzzy transforms," *International Journal of Approximate Reasoning*, vol. 48, pp. 110–131, 2008.
[20] V. Loia and S. Sessa, "Fuzzy relation equations for coding/decoding processes of images and videos," *Information Sciences*, vol. 171, pp. 145–172, 2005.
[21] A. Kanemura, S. Maeda, and S. Ishii, "Sparse bayesian learning of filters for efficient image expansion," *IEEE Transactions on Image Processing*, vol. 19, pp. 1480–1490, 2010.
[22] A. Jurio, M. Pagola, R. Mesiar, G. Beliakov, and H. Bustince, "Image magnification using interval information," *IEEE Transactions on Image Processing*, vol. in press, doi:10.1109/TIP.2011.2158227, 2011.
[23] H. Bustince, E. Barrenechea, and M. Pagola, "Restricted equivalence functions," *Fuzzy Sets and Systems*, vol. 157, pp. 2333–2346, 2006.
[24] H. Bustinces, E. Barrenechea, and M. Pagola, "Relationship between restricted dissimilarity functions, restricted equivalence functions and normal e-n-functions: Image thresholding invariant," *Pattern Recognition Letters*, vol. 29, pp. 525–536, 2008.
[25] H. Bustince, V. Mohedano, E. Barrenechea, and P. M, "Definition and construction of fuzzy di-subsethood measures," *Information Sciences*, vol. 176, pp. 3190–3231, 2006.
[26] H. Bustince, M. Pagola, and E. Barrenechea, "Construction of fuzzy indices from fuzzy di-subsethood measures: Application to the global comparison of images," *Information Sciences*, vol. 177, pp. 906–929, 2007.

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.

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.

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.