Ordered directional monotonicity in the construction of edge detectors

C. Marco-Detchart[a,b,c], H. Bustince[a,b,c], J. Fernandez[a,b,c], R. Mesiar[e,f], J. Lafuente[a], E. Barrenechea[a,b,c], J. M. Pintor[d]

[a] *Departamento de Estadistica, Informatica y Matematicas, Universidad Publica de Navarra, Campus de Arrosadia, 31006, Pamplona, Spain*
[b] *Institute of Smart Cities, Universidad Publica de Navarra, Campus de Arrosadia, 31006, Pamplona, Spain*
[c] *Laboratory Navarrabiomed, Hospital Complex of Navarre (CHN), Universidad Publica de Navarra, IdiSNA, Irunlarrea 3. 31008, Pamplona, Spain*
[d] *Departamento de Ingeniería, Universidad Publica de Navarra, Campus de Arrosadia, 31006, Pamplona, Spain*
[e] *Institute of Information Engineering, Automation and Mathematics, Slovak University of Technology, Radlinskeho 11, 810 05 Bratislava, Slovakia*
[f] *Palacky University Olomouc, Dept. Algebra and Geometry, Faculty of Sciences, Olomouc, Czech Republic*

Abstract

In this paper we provide a specific contruction method of ordered directionally monotone functions. We show that the functions obtained with this construction method can be used to build edge detectors for grayscale images. We compare the results of these detectors to those obtained with some other ones that are widely used in the literature. Finally, we show how a consensus edge detector can be built improving the results obtained both by our proposal and by those in the literature when applied individually.

Keywords: Ordered directionally monotone function; Directional monotonocity; Edge detection; Consensus image.

1. Introduction

In recent years, the analysis of functions which fulfill monotonicity conditions weaker than those required to aggregations is attracting a growing interest. The study of monotonicity has led to the development of new concepts as, for instance, weak monotonicity (where monotonicity is required only along the ray defined by the vector $(1, \ldots, 1)$ [42]) or directional monotonicity (where monotonicity is required along some ray defined by a vector in the first quadrant [11]). These extensions have shown to be very useful in different application fields, as in classification problems [31].

A common feature shared by all these extensions is that the monotonicity direction is the same for every considered input. But, in some applied problems, this can be too restrictive.

Let us consider the problem of automatic object identification on images [3, 17, 24, 25, 32]. One of the most important steps for object identification consists in extracting the object edges in the image [10]. Recall that the contours of the visible objects are denoted as edges if there exists a big enough jump between the intensity of a pixel and those of its neighbours. Clearly, this is an imprecise definition, but this is due to the inherent uncertainty of the border concept itself. [28].

Many edge detection methods use, among other techniques, the *gradient* vector to represent intensity jumps between pixels [37, 41], as for example the well-known approaches by Sobel and Feldman [40], Prewitt [38] and Canny [15], being the latter still considered as one of the most important references for comparison in order to determine the quality of a given method. In recent years, different methods which use Machine

Email addresses: cedric.marco@unavarra.es (C. Marco-Detchart), bustince@unavarra.es (H. Bustince), fcojavier.fernandez@unavarra.es (J. Fernandez), mesiar@math.sk (R. Mesiar), lafuente@unavarra.es (J. Lafuente), edurne.barrenechea@unavarra.es (E. Barrenechea), txma@unavarra.es (J. M. Pintor)

Learning techniques have appeared, as the ones based on Random Forest [16], or Newton's Gravitational Law, as for instance, [28], etc. Furthermore, deep learning methods [43, 44, 45] have recently become the most widely used ones due to their high performance, which improves all the previously mentioned algorithms. Nevertheless, deep learning techniques have a high computational cost and require a previous training, so in some cases it is enough to use simpler, less costly methods.

In order to tackle our problem, we stick to the Bezdek Breakdown Structure for edge detection [4], a framework that considers four phases for edge detection tasks:

($S1$) Conditioning: Applying a Gaussian filter (with $\sigma_G = 2$) to the image I obtaining a new smoothed image, I_G.

($S2$) Feature extraction: Obtaining a new image I_M from I_G, where each pixel in I_M represents information about the intensity jump between the corresponding pixel and its neighbours in I_G.

($S3$) Blending: Using the non-maximum suppression procedure [15], computing prior orientations by Kovesis' function [27], obtaining a thinned image, I_H.

($S4$) Scaling: Applying an hysteresis method [36] to obtain the binary edge image I_B. Such representations are commonly demanded to every edge detection approach fulfilling the restrictions imposed by Canny [14, 15].

Focusing our attention on the *feature extraction* phase ($S2$), we observe that the information provided by a pixel and its neighbours is to be considered simultaneously. But, in order not to loose information all the intensity differences have to be considered and not only the difference between the maximal and the minimal intensities [6] Furthermore, the relevant information must consider all the intensity jumps in the neighbourhood and their mutual relation based on their relative size. The nature of edges implies that the intensity jumps must be ordered in a decreasing way, as the bigger an intensity jump is, the larger its influence is in the possible existence of an edge. In this sense, it is straightforward to consider the directions defined by the increasingness or decreasingness of the intensity, but such directions may vary from one pixel to another one. For this reason, we consider the use of ordered directionally monotone functions as a way of issuing this problem.

Bearing in mind all the previous considerations, we aim at the following goals in this work:

1. Develop a new mathematical method to build ordered directionally monotone (ODM) functions.
2. Show that the new building method for ODM functions can be useful to build a feature image extractor for phase ($S2$) of the edge detection breakdown structure.

Besides, as we have already commented previously, there exists many different methods to get the feature image in phase ($S2$). For this reason, we consider the following goal to complete our experimental study:

3. Build, using penalty functions [13, 8, 9], a consensus feature image extractor from the different algorithms used in the experimental setup of phase ($S2$).

The structure of this work is as follows. In Section 2 some of the basic notation and concepts needed for our proposal are exposed. Section 3 is devoted to expose a specific case of OD monotone function and Section 4 develops the previous section ideas in the context of image feature extraction along with two alternative constructions (Section 5). Section 6 presents a consensus edge detector build upon the combination of the different considered methods. The experimental framework is exposed in Section 7. Then, Section 8 shows the quantitative evaluation of the proposed methods compared with different alternatives of the literature. Finally in Section 9 some conclusions and future works are presented.

2. Preliminaries

2.1. Basic notations and concepts

In this subsection we fix some notations and concepts which will be useful for the remainder of the work.

Let $n > 1$. We use bold letters to denote points in the hypercube $[0,1]^n$, i.e., $\mathbf{x} = (x_1, \ldots, x_n) \in [0,1]^n$. In particular, we write $\mathbf{0} = (0, \ldots, 0)$ and $\mathbf{1} = (1, \ldots, 1)$. Given $\mathbf{x}, \mathbf{y} \in [0,1]^n$ we write $\mathbf{x} \leq \mathbf{y}$ if $x_i \leq y_i$ for every $i \in \{1, \ldots, n\}$. Note that this relation is a partial order which extends the usual linear order between real numbers.

For $n > 1$, we denote by P_n the set of permutations of $\{1, \ldots, n\}$. That is,

$$P_n = \{\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\} \mid \sigma \text{ is bijective}\}.$$

Given $\sigma \in P_n$, $\mathbf{x} \in [0,1]^n$ and $\vec{r} \in \mathbb{R}^n$, we define:

$$\mathbf{x}_\sigma = (x_{\sigma(1)}, \ldots, x_{\sigma(n)})$$

and

$$\vec{r}_\sigma = (r_{\sigma(1)}, \ldots, r_{\sigma(n)}).$$

For simplicity, in this work we refer as fusion function (of dimension n) to any function $F : [0,1]^n \to [0,1]$.

A distinguished class of fusion functions, mainly used in applied tasks, is that of aggregation functions [20, 2].

Definition 1. *[1, 12] A mapping $M : [0,1]^n \to [0,1]$ is an aggregation function if it is monotone non-decreasing in each of its components and satisfies $M(\mathbf{0}) = M(0, 0, \ldots, 0) = 0$ and $M(\mathbf{1}) = M(1, 1, \ldots, 1) = 1$.*

2.2. Ordered directional monotonicity

Imposing monotonicity might be too restrictive for some specific applications (e.g., the mode is not increasing with respect to all its arguments but it is a valid function for certain applications). This consideration led Wilkin and Beliakov [42] to introduce the notion of weak monotonicity.

This concept of weak monotonicity can be further extended by the notion of directional monotonicity, that we define now.

Definition 2. *[11] Let $\vec{r} = (r_1, \ldots, r_n)$ be a real n-dimensional vector, $\vec{r} \neq \vec{0}$. A fusion function $F : [0,1]^n \to [0,1]$ is $\vec{r}$-increasing if for all points $(x_1, \ldots, x_n) \in [0,1]^n$ and for all $c > 0$ such that $(x_1 + cr_1, \ldots, x_n + cr_n) \in [0,1]^n$ it holds*

$$F(x_1 + cr_1, \ldots, x_n + cr_n) \geq F(x_1, \ldots, x_n) \, .$$

That is, an $\vec{r}$-increasing function is a function which is increasing along the oriented segments determined by the vector $\vec{r}$. For this reason, we say that F is directionally monotone, or, more specifically, directionally $\vec{r}$-increasing. For an in-depth study of the concept of directional monotonicity see [11]. Nevertheless, it is worth to mention that directional monotonicity combined with appropriate boundary conditions leads to the notion of pre-aggregation function, see [31].

In [7] the notion of ordered directionally monotone function (ODM) is presented. To motivate its introduction, note that by means of directional monotonicity, usual monotonicity may be relaxed, just requiring increasingness along some fixed ray. However, the direction along which monotonicity is demanded is the same for every point in the domain $[0,1]^n$, and it is independent of the particular point which is being considered.

For ODM functions, on the contrary, the direction along which monotonicity is required varies depending on the relative size of the coordinates of the considered input. The formal definition reads as follows.

Definition 3. *[7] Let $F : [0,1]^n \to [0,1]$ be a fusion function and let $\vec{r} \neq \vec{0}$ be a n-dimensional vector. F is said to be ordered directionally (OD) $\vec{r}$-increasing if for any $\mathbf{x} \in [0,1]^n$, for any $c > 0$ and for any permutation $\sigma \in P_n$ with $x_{\sigma(1)} \geq \cdots \geq x_{\sigma(n)}$ and such that*

$$1 \geq x_{\sigma(1)} + cr_1 \geq \cdots \geq x_{\sigma(n)} + cr_n \geq 0,$$

it holds that

$$F(\mathbf{x} + c\vec{r}_{\sigma^{-1}}) \geq F(\mathbf{x}),$$

where $\vec{r}_{\sigma^{-1}} = (r_{\sigma^{-1}(1)}, \ldots, r_{\sigma^{-1}(n)})$.

Analogously, F is said to be OD $\vec{r}$-decreasing if for any $\mathbf{x} \in [0,1]^n$, for any $c > 0$ and for any permutation $\sigma \in P_n$ with $x_{\sigma(1)} \geq \cdots \geq x_{\sigma(n)}$ and such that

$$1 \geq x_{\sigma(1)} + cr_1 \geq \cdots \geq x_{\sigma(n)} + cr_n \geq 0,$$

it holds that

$$F(\mathbf{x} + c\vec{r}_{\sigma^{-1}}) \leq F(\mathbf{x}),$$

where $\vec{r}_{\sigma^{-1}} = (r_{\sigma^{-1}(1)}, \ldots, r_{\sigma^{-1}(n)})$.

By an OD $\vec{r}$-monotone function we mean a function which is OD $\vec{r}$-increasing or ordered directionally $\vec{r}$-decreasing.

Note that that for symmetric fusion functions, both concepts of directional monotonicity and ordered directional monotonicity coincide, and then, in particular, such fusion functions are weakly increasing in the sense of Wilkin and Beliakov.

Example 4. *The weighted Lehmer mean*

$$L_\lambda(x,y) = \frac{\lambda x^2 + (1-\lambda)y^2}{\lambda x + (1-\lambda)y},$$

with the convention $0/0 = 0$, is $(1-\lambda, \lambda)$-increasing. It follows that the function

$$G_\lambda(x,y) = \frac{\lambda(\vee(x,y))^2 + (1-\lambda)(\wedge(x,y))^2}{\lambda \vee (x,y) + (1-\lambda) \wedge (x,y)}$$

is OD $(\lambda, 1-\lambda)$-increasing. Note that, if $\lambda \in]0,1[$, then L_λ is a pre-aggregation function which is not an aggregation function, see [31].

3. A particular case of OD monotone functions

In this section we discuss an affine construction method for ODM functions.

Theorem 5. *Let $G : [0,1]^n \to [0,1]$ be defined, for $\mathbf{x} \in [0,1]^n$ and $\sigma \in P_n$ such that $x_{\sigma(1)} \geq \ldots \geq x_{\sigma(n)}$, by*

$$G(\mathbf{x}) = a + \sum_{i=1}^{n} b_i x_{\sigma(i)},$$

for some $a \in [0,1]$ and $\vec{b} = (b_1, \ldots, b_n) \in \mathbb{R}^n$ such that $0 \leq a + b_1 + \cdots + b_j \leq 1$ for all $j \in \{1, \ldots, n\}$. Then G is OD $\vec{r}$-increasing for every non-null vector $\vec{r}$ such that $\vec{b} \cdot \vec{r} \geq 0$. In particular, for every non-null vector $\vec{r}$ which is orthogonal to $\vec{b}$.

Proof.

Take $\sigma \in P_n$ such that $x_{\sigma(1)} \geq x_{\sigma(2)} \geq \cdots \geq x_{\sigma(n)}$. Take also $c > 0$ such that $(x_{\sigma(1)} + cr_1, \ldots, x_{\sigma(n)} + cr_n) \in [0,1]^n$. Then we have that

$$\begin{aligned} G(\mathbf{x} + c\vec{r}_{\sigma^{-1}}) &= a + \sum_{i=1}^{n} b_i (x_{\sigma(i)} + cr_i) \\ &= a + \sum_{i=1}^{n} b_i x_{\sigma(i)} + c \sum_{i=1}^{n} b_i r_i \\ &\geq G(\mathbf{x}), \end{aligned}$$

as we wanted to show. ∎

Theorem 5 can be generalized taking into account the following lemma.

Lemma 6. *Let $\varphi : [0,1] \to [0,1]$ be an automorphism (i.e., an increasing bijection). Then, if $G : [0,1]^n \to [0,1]$ is an ordered directionally increasing function, the function $\varphi \circ G$ is also an ordered directionally increasing function.*

Proof.

It follows straightforwardly from the definition of ordered directionally increasing functions and the fact that φ is increasing. ∎

From Lemma 6, we have the following corollary of Theorem 5, which is very relevant for our edge detectors, since the value of p allows us to darken or lighten the considered image.

Corollary 7. *Let $p > 0$. Let $G : [0,1]^n \to [0,1]$ be defined, for $\mathbf{x} \in [0,1]^n$ and $\sigma \in P_n$ such that $x_{\sigma(1)} \geq \ldots \geq x_{\sigma(n)}$, by*

$$G(\mathbf{x}) = \left(a + \sum_{i=1}^{n} b_i x_{\sigma(i)} \right)^{\frac{1}{p}}, \tag{1}$$

for some $a \in [0,1]$ and $\vec{b} = (b_1, \ldots, b_n) \in \mathbb{R}^n$ such that $0 \leq a + b_1 + \cdots + b_j \leq 1$ for all $j \in \{1, \ldots, n\}$. Then G is OD $\vec{r}$-increasing for every non-null vector $\vec{r}$ such that $\vec{b} \cdot \vec{r} \geq 0$.

Proof.

It follows from Lemma 6 taking into account that the function $\varphi(x) = x^{\frac{1}{p}}$ is an automorphism. ∎

The following result is very relevant from the point of view of applications in image processing, since it can be straightforwardly applied to the cases in which an edge exists or not.

Corollary 8. *Let $p > 0$ and let $G : [0,1]^n \to [0,1]$ be defined as in Corollary 7. Then*

(i) $G(\mathbf{0}) = 0$ if and only if $a = 0$.

(ii) Assume that $a = 0$. Then, $G(\mathbf{1}) = 1$ if and only if $b_1 + \cdots + b_n = 1$.

Proof. To see (i), observe that $G(\mathbf{0}) = a^{\frac{1}{p}}$, so the result is straightforward. Regarding (ii), $G(\mathbf{1}) = (\sum_{i=1}^{n} b_i \cdot 1)^{\frac{1}{p}}$, and the result follows. ∎

4. A new algorithm to construct a feature image using Ordered Directionally Monotone functions

In Algorithm 1 we present the process to obtain a feature image by means of ODM functions, phase $(S2)$, which by definition consider both the intensity jumps between a pixel and its neighbours and the direction along which such jumps vary, in a decreasing way.

Algorithm 1 Algorithm to construct a feature image using ODM functions

Input: A normalized greyscale image I_G and a parameter $p > 0$ to build an ODM function G as in Corollary 7.

Output: A feature image I_M.

1: **for** each pixel (x, y) of I_G **do**
2: Compute the 8 values obtained by applying the absolute value of the difference between $I_g(x, y)$ and its 8-neighbourhood;
3: Order the eight values of step 2 in a decreasing way;
4: Fix the parameters a, $\vec{r}$ y $\vec{b}$ according to the vector obtained in step 3.
5: Build the ODM function G as in Corollary 7 with the parameters obtained in step 4.
6: Apply the ODM function G to the values obtained in step 3;
7: Assign as intensity of the pixel (x, y) of I_M the value obtained in step 6.
8: **end for**

Next, let us expose how to carry out the execution of the Algorithm 1. Firstly, let us consider that the pixel (x, y) of I_G is the pixel a_{22} of Fig. 1 and then compute the 8 values indicated in step 2, obtaining the following outcome:

$$x_1 = |a_{22} - a_{11}|,\ x_2 = |a_{22} - a_{12}|,$$
$$x_3 = |a_{22} - a_{13}|,\ x_4 = |a_{22} - a_{23}|,$$
$$x_5 = |a_{22} - a_{33}|,\ x_6 = |a_{22} - a_{32}|,$$
$$x_7 = |a_{22} - a_{31}|,\ x_8 = |a_{22} - a_{21}|.$$

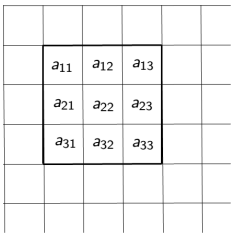


Figure 1: Pixel a_{22} and its 8-neighbourhood.

In step 3, these differences are ordered in a decreasing way; that is,

$$x_{\sigma_{(1)}} \geq x_{\sigma_{(2)}} \geq \ldots \geq x_{\sigma_{(7)}} \geq x_{\sigma_{(8)}}.$$

Note that, as we have already said, there exists an edge if there is a big enough intensity jump between a pixel and its neighbours. So the greatest intensity differences are the most relevant ones in order to determine if there is an edge or not.

In step 4 we should fix the parameters a, $\vec{r}$ y $\vec{b}$, further explained in Section 5. It does not exist a precise procedure to obtain these parameters, but nevertheless in Section 5 we discuss two possible choices of such parameters. Both of them are based on the expected properties of edges. These parameters are necessary to get in step 5 an ODM function G as in Corollary 7. Finally, in step 6, the ODM function G is applied to get the feature image.

5. Two alternative sets of parameters for the ODM function construction

In this section we discuss two expressions for ODM functions for Algorithm 1. These expressions are

obtained from Corollary 7 using Eq. (1) and giving specific values to the parameters a, p, vector $\vec{r}$ and $\vec{b}$. It is important to remark that these expressions are a first approach and they have not been optimized. On the contrary, we discuss them due to their simplicity. In a future work, we intend to optimize the value of the different parameters depending on the specific type of images that we consider.

Observing the expression given in Eq. (1) for step 6 in Algorithm 1, the parameter p allows us to darken or lighten the resulting feature image. It is enough to observe that if $p > 1$, then we get a lighter feature image, and if $0 < p < 1$, then we get a darker one (see [18]).

5.1. Case 1

We consider:

$$\vec{r} = (x_{\sigma_{(1)}}, x_{\sigma_{(2)}}, x_{\sigma_{(3)}}, x_{\sigma_{(4)}}, x_{\sigma_{(5)}}, x_{\sigma_{(6)}}, x_{\sigma_{(7)}}, x_{\sigma_{(8)}});$$

$$\vec{b} = \begin{cases} \left(\frac{x_{\sigma_{(1)}}}{\sum_{i=1}^{8} x_{\sigma_{(i)}}}, \ldots, \frac{x_{\sigma_{(7)}}}{\sum_{i=1}^{8} x_{\sigma_{(i)}}}, \frac{x_{\sigma_{(8)}}}{\sum_{i=1}^{8} x_{\sigma_{(i)}}} \right) & \text{if } \sum_{i=1}^{8} x_{\sigma_{(i)}} \neq 0 \\ (0, \ldots, 0) & \text{otherwise.} \end{cases}$$

$a = 0$ and $\frac{1}{p} = 0.30$.

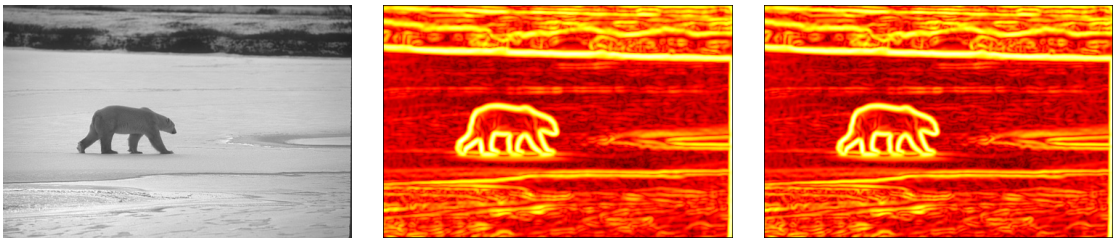
Regarding $\vec{r}$, the highest value , $x_{\sigma(1)}$, is the most relevant for the possible edge, since it corresponds to the biggest intensity jump between the central pixel and its neighbours. With respect to the value of a, we take $a = 0$ because if all the values to be aggregated are null, i.e., if $x_{\sigma(1)} = \cdots = x_{\sigma(8)} = 0$, this means that in the considered 8-neighbourhood all the pixels have the same intensity and hence there is no edge. So, in this case, we should have $G(0, \ldots, 0) = 0$, and from Corollary 8, this is so if and only if $a = 0$. Besides, if $x_{\sigma(1)} = \cdots = x_{\sigma(8)} = 1$, the intensity jump between the pixels in the 8-neighbourhood and the central pixel is as large as possible. So $G(1, \ldots, 1)$ must be equal to 1 and, again from Corollary 8, since $a = 0$, we should require that $b_1 + \cdots + b_n = 1$.

Regarding the computation of the parameter $1/p$ for constructing the ODM functions, according to Corollary 7, we do as follows: We consider the points $1/p_1^1 = 0.1$, $1/p_1^2 = 0.2$, $1/p_1^3 = 0.3$,..., $1/p_1^9 = 0.9$ (i.e., a uniform partition of the interval $[0, 1]$). We apply Algorithm 1 with each of these nine values and we evaluate the quality of the resulting edge images (in terms of the average of the values of $F_{0.5}$ on the considered training images). Let i_1 be the index such that, if we apply Algorithm 1 with $1/p = 1/p_1^{i_1}$, we get the best result among the nine considered values. Denote $1/p_1^{fix} = p_1^{i_1}$. Next, consider the interval $[1/p_1^{fix} - 0.05, 1/p_1^{fix} + 0.05]$. We take again a uniform partition of this interval with nine points, $1/p_2^1 = 1/p_1^{fix} - 0.04$, $1/p_2^2 = 1/p_1^{fix} - 0.03$, ... ,$1/p_2^9 = 1/p_1^{fix} + 0.04$ and we repeat again the procedure of running and evaluating the results of Algorithm 1 for each of these 9 values of the parameter $1/p$ to get a new point $1/p_2^{fix}$. We repeat the procedure n times, where the new interval around the point $1/p_i^{fix}$ is $[1/p_i^{fix} - 5 \times 10^{-i}, 1/p_i^{fix} + 5 \times 10^{-i}]$ $(i = 2, \ldots, n)$ and we finish when the variation of the average value of the measure $F_{0.5}$ between the images obtained after applying Algorithm 1 with $1/p = 1/p_n^{fix}$ and those obtained with $1/p = 1/p_{n+1}^{fix}$ is smaller than a given tolerance, in our case 0.001.

Following this procedure with the training images in the considered dataset, we get the cited value $1/p = 0.30$. Finally, observe that the expression that we are actually recovering in this situation is $G(\mathbf{x}) = \sum_{i=1}^{8} \frac{x_i^2}{\sum_{i=1}^{8} x_i}$. However, this simple expression is justified by the previous considerations.

5.2. Case 2

In this case we consider:



(a) *Original* (b) *Case 1* (c) *Case 2*

Figure 2: Original image from BSDS [33] (100007) along with feature images obtained after applying Algorithm 1 with Case 1 and case 2.

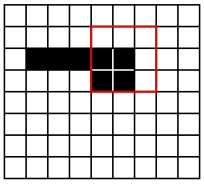


Figure 3: Example of a 3×3 neighbourhood near an edge. White pixels get value 0 and black pixels 1.

$$\vec{r} = (x_{\sigma(1)}, x_{\sigma(2)}, x_{\sigma(3)}, x_{\sigma(4)}, x_{\sigma(5)}, x_{\sigma(6)}, x_{\sigma(7)}, x_{\sigma(8)});$$
$$\vec{b} = \left(\frac{1}{8}\left(1 - \left|x_{\sigma(1)} - \underset{i\in\{1,\ldots,8\}}{\text{median}}\{x_i\}\right|\right), \ldots \right.$$
$$\left. \ldots, \frac{1}{8}\left(1 - \left|x_{\sigma(8)} - \underset{i\in\{1,\ldots,8\}}{\text{median}}\{x_i\}\right|\right)\right);$$

$a = 0$ and $\frac{1}{p} = 0.30$. The justification for the choice of these parameters is analogous to that in Case 1.

Note that in this case necessarily $a = 0$. Indeed, it suffices to consider the case $x_1 = \ldots = x_8$.

In Fig. 2 we show the results obtained by applying Algorithm 1 with the two ODM functions, Case 1 and Case 2, to an original image, Fig. 2a.

As an example of a simple case where pixels are one of the possible extremes of the set (white pixels as 0 and black pixels as 1) we take the 3×3 neighbourhood in red of Figure 3 and show the results obtained with each one of the proposed alternatives. Considering the previous explanations we have:

$$\vec{r} = (1, 1, 1, 1, 1, 0, 0, 0);$$
$$\text{Case 1 } \vec{b} = \left(\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, 0, 0, 0\right);$$
$$\text{Case 2 } \vec{b} = \left(\frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, 0, 0, 0\right)$$

$$\vec{r} = (1,1,1,1,1,0,0,0);$$

Case 1 $\vec{b} = \left(\frac{1}{5},\frac{1}{5},\frac{1}{5},\frac{1}{5},\frac{1}{5},\frac{1}{5},0,0,0\right)$;

Case 2 $\vec{b} = \left(\frac{1}{8},\frac{1}{8},\frac{1}{8},\frac{1}{8},\frac{1}{8},\frac{1}{8},0,0,0\right)$

$$\text{Case 1 } G(\mathbf{x}) = \left(a + \sum_{i=1}^{n} b_i x_{\sigma(i)}\right)^{\frac{1}{p}} = (0+1)^{\frac{1}{p}} = 1;$$

$$\text{Case 2 } G(\mathbf{x}) = \left(a + \sum_{i=1}^{n} b_i x_{\sigma(i)}\right)^{\frac{1}{p}} = (0+1)^{\frac{1}{p}} = 1$$

As we can see both of the alternatives permit to extract the features of an image and hence be used in edge extraction.

6. Consensus feature images

We have seen in the previous sections that, from a given image, we can build different feature images in phase (S2). In this section we discuss a method to build a consensus feature image from the different feature images. This consensus image can be built using aggregation function pixel by pixel, and there are several ways of doing it. Given a set of aggregation functions $M_1, \ldots, M_s$, we may:

1. fFx one of this aggregation functions, M_i, and apply it to the intensities of the first pixel in every feature image to get the intensity of the first pixel in the consensus image. Then we can apply it again to the intensities of the second pixel in every feature image to get the intensity of the second pixel in the feature image, and we can repeat the process, always with the same M_i, for the intensities of all the other pixels; or we may
2. Use the notion of penalty function [5], which allows us to choose the best aggregation (among the s considered ones) for each pixel. In this case, it may happen that, in order to build the intensity of the first pixel in the consensus image, we use an aggregation function M_j, whereas to build the intensity of the second pixel we use an aggregation function M_k different from M_j, and so on.

The advantage of using penalty functions is that the aggregation used to build the intensity of each pixel in the consensus image is the one which provides the least dissimilar result from the intensities of the corresponding pixels in the feature images. So it seems logical to use penalty functions to build the consensus feature image. But the main problem with this method is the choice of the best penalty function. In our experimentation we will use the following expression (see [5]):

$$P_{\nabla}(\mathbf{X},\mathbf{y}) = \sum_{q=1}^{m}\sum_{p=1}^{n} |x_p^q - y_q|^2 \tag{2}$$

where we have n feature images to aggregate, each of them with m pixels. In this way, x_p^q denotes the intensity of the pixel q in image p and y_q is the result of aggregating $x_1^q, \ldots, x_n^q$ by means of some of the considered aggregation functions. Finally, we denote $\mathbf{X} = ((x_1^1, x_2^1, \ldots, x_n^1), \ldots, (x_1^m, x_2^m, \ldots, x_n^m))$ and $\mathbf{y} = (y_1, \ldots, y_m)$.

We use Eq. 2 since it is among the most used ones in image comparison, as it is based on the mean squared error; and, moreover, due to the following property.

If the arithmetic mean is among the considered aggregation functions $M_1, \ldots, M_s$ set then the least dissimilar result is obtained applying in every case the arithmetic mean.

To analyse, this property, let us recall the concept of penalty function in a Cartesian product of lattices, that was deeply studied in [5]. Consider the following goal: *given a set of n numerical values $x_1, \ldots, x_n$ and q averaging aggregation (i.e., between the minimum and the maximum, see [2]) functions $M_1, \ldots, M_q$ penalty functions (see [13, 8, 9]) allow us to select, between the q functions, the one that provides the output least dissimilar to all the inputs. That is, we choose the aggregation functions using a consensus procedure based on testing several functions until we find the one providing the least dissimilar result with respect to the values of the inputs.*

The definition of penalty function in a Cartesian product of lattices reads as follows [8]:

Definition 9. *For any closed interval $\mathbb{I} \subseteq \mathbb{R}$, the function $P : \mathbb{I}^{n+1} \to \mathbb{R}^+$ is a penalty function if and only if there exists $c \in \mathbb{R}^+$ such that:*

1. *$P(\mathbf{x}, y) \geq c$, for all $\mathbf{x} \in \mathbb{I}^n, y \in \mathbb{I}$;*
2. *$P(\mathbf{x}, y) = c$ if and only $x_i = y$, for all $i = 1 \ldots n$, and*
3. *P is quasi-convex lower semi-continuous in y for each $\mathbf{x} \in I_n$.*

To understand how the penalty function works, let us consider the following example:

Let us assume that we have three feature images A, B, C, obtained with three different methods in phase (S2).

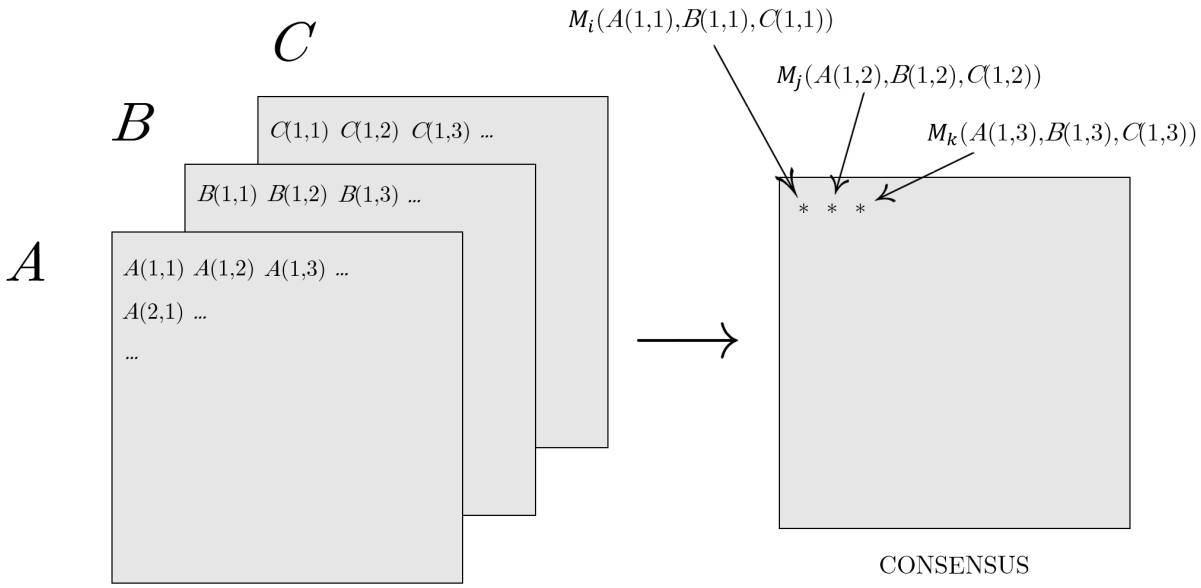


Figure 4: Consensus image.

We want to choose, among a set of aggregation functions $M_1, \ldots, M_s$, the combinations of three aggregation functions M_i, M_j y M_k which provide the smallest value of the considered penalty function.

Looking at Fig. 4, where M_i, M_j and M_k are three aggregation functions, it is clear that inputs can be seen as vectors, so we should use penalty functions defined over a Cartesian product of lattices. That is, functions defined as in Fig. 5, where it is clear that each combination of three aggregation functions M_i, M_j y M_k gives us a value P_{ijk}. so the idea is to determine which combination of three aggregation functions provides the smallest value of P_{ijk} for the considered pixels.

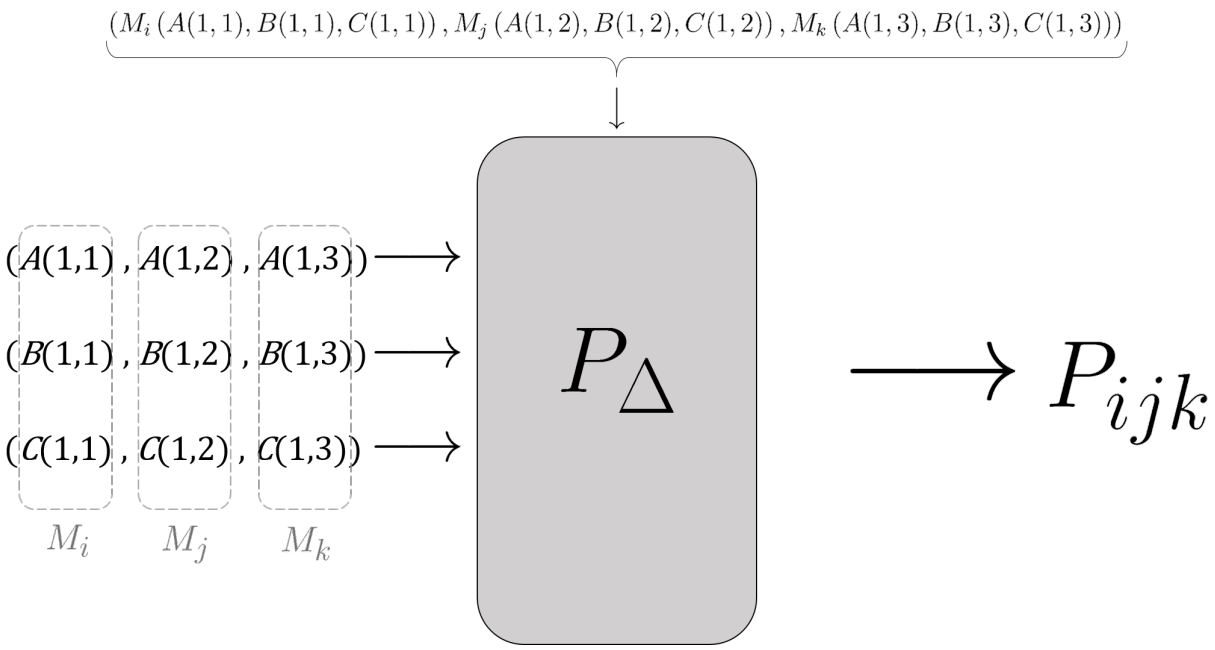


Figure 5: Penalty function

As we have said, for the experimentation we use the following specific expression:

$$P_\nabla(\mathbf{X},\mathbf{y}) = \sum_{q=1}^{m}\sum_{p=1}^{n}|x_p^q - y_q|^2. \tag{3}$$

In the literature, the notion of penalty based function [13] or P-function [8] (for short) is used to refer to the function which selects the value y which minimizes the value of the penalty function for any $(x_1,\ldots,x_n)$, i.e.,

$$f(x_1,\ldots,x_n) = \underset{y}{\operatorname{argmin}}\ P(x_1,\ldots,x_n,y)$$

if y is the unique minimizer and $y = \frac{(a+b)}{2}$ if the set of minimizers is the interval (a,b) (open or closed). It is also known that in some cases the penalty based function can be expressed analytically, while in other cases it is not possible. For the proposed penalty function of the experimentation given in Eq. (2), we have the following result.

Theorem 10. *Let P be the penalty function given in Eq. (2). The penalty based function of P can be expressed analytically and its expression is the arithmetic mean, i.e., the value of y which minimizes the penalty function is*

$$f(x_1,\ldots,x_n) = P(x_1,\ldots,x_n,\frac{x_1+\ldots+x_n}{n})$$

Proof. It is straight taking into account that the arithmetic mean is the P-function of a given $P(x_1,\ldots,x_n,y) = \sum_{i=1}^{n}(x_i-y)^2$ (see for example [8]). ∎

So, with the considered expression, the least dissimilar result is obtained using in every case the arithmetic mean, i.e., for $\mathbf{y} = (\frac{1}{n}\sum_{i=1}^{n}x_i^1,\ldots,\frac{1}{n}\sum_{i=1}^{n}x_i^m)$.

Taking into account this property and the results in the Annex, in this work we use the arithmetic mean to build the consensus feature image.

We leave for future works the analysis of other possible expressions for the penalty functions, and hence, of different aggregation functions.

7. Experimental framework

In this section, we present the set-up of the experimental framework used to develop the empirical comparison in this work. Given a grayscale image I, we consider it as a matrix of elements (pixels) arranged in rows and columns, where each pixel takes an intensity value in $\{0, 1, \ldots, L-1\}$. A prior step is to normalize the intensities to values belonging to $[0, 1]$. As we have said, our proposal to perform the experiments for a given grayscale image I, considering Bezdek *et al.* [4] includes phases $(S1)$-$(S4)$ described in the introduction. Moreover, we also need to evaluate the quality of the edges that we obtain, so we also consider the following fifth step to compare to ground truth images, i.e,.

$(S5)$ Edge image evaluation. Compare the binary edge image I_B with hand-labeled segmentations, getting measurements in terms of *Precision* $(PREC)$, *Recall* (REC) and $F_{0.5}$ measure [29].

7.0.1. The dataset

For our experiments we have used the images of Berkeley Segmentation Dataset (BSDS500) [33], specifically 200 test natural images in grayscale with dimensions of 321×481 or 481×321. Associated to each original image there exist several hand-labeled segmentations (done by humans), which, as we have already said, are denoted as *ground truth images*; usually there are between 4 and 9 ground truth images associated to each original image (see Fig. 6). In this sense, the ground truth images are considered as ideal images and they serve to test if the results obtained by an edge detection method are similar, or not, to the ones segmented by a human.

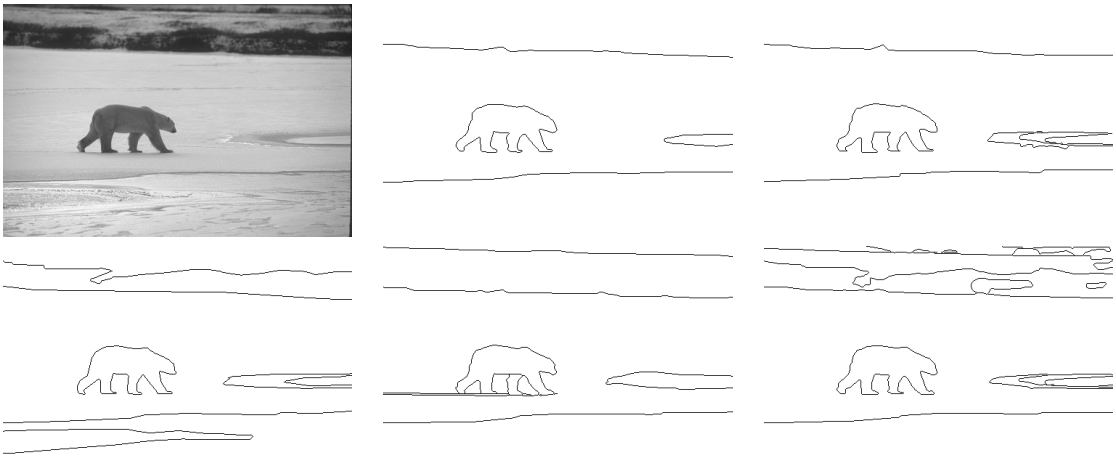


Figure 6: The original image and its five ground truth images (BSDS500).

7.0.2. Edge detection approaches and parameter configuration

Firstly, in step $(S1)$ all the approaches perform a Gaussian filter with standard deviation $\sigma_G = 2$. Secondly, to obtain a feature image, $(S2)$, we consider the following edge detection approaches: the Canny method [15], gravitational approach [28, 30], fuzzy morphology approach [21], Structured Forests method [16] and the one based on ODM functions (Algorithm 1); all these methods have different configurations to be executed, which are listed below:

- The Canny method [15]. We obtain the magnitude of the gradient estimation with the convolution operator proposed by Canny, with $\sigma_C = 2.25$ which is a common value considering the size of the images [23, 30, 35]. This σ_C is different from that required for calculating the Gaussian filter, i.e.,

σ_G. Note that the unsupervised calculation of an optimal value of σ_C has not been solved so far. We denote this approach by *C*.

- Gravitational [28, 30]. This approach is based on gravitational forces using relief functions where, as particular cases, t-norms and t-conorms are used. In our experiments, we denote by G_{S_P} and G_{S_M}, when the probabilistic sum ($S_P(x,y) = x + y - xy$) and the maximum ($S_M(x,y) = \vee(x,y)$) are considered as t-conorms, respectively.

- Fuzzy Morphology [21, 22]. The authors proposed a generalization of the morphological operators based on considering general t-conorms and t-norms in erosion and dilation definitions. We have carried out in our experiments the best configurations claimed in [21, 22]: for erosion the Kleene-Dienes implication operator (Eq. (4)) and for dilation, the nilpotent minimum as t-norm (Eq. (5)).

$$I_{KD}(x,y) = \vee(1-x,y); \tag{4}$$

$$T_{nM}(x,y) = \begin{cases} 0, & \text{if } x+y \leq 1, \\ \wedge(x,y) & \text{otherwise.} \end{cases} \tag{5}$$

  Considering the pair (I_{KD}, T_{nM}) we use the best two parameterizations given in [21, 22]:

  – FM_{SS}. The Schweizer-Sklar t-norm (T_λ^{SS}) [39] is applied to Eq. (4) to get the fuzzy erosion and for fuzzy dilation the Schweizer-Sklar t-conorm (defined as the dual of Schweizer-Sklar t-norm) is applied to Eq. (5).

$$T_\lambda^{SS}(x,y) = \begin{cases} \wedge(x,y), & \text{if } \lambda = -\infty, \\ xy & \text{if } \lambda = 0, \\ T_D(x,y) & \text{if } \lambda = +\infty, \\ (\vee(x^\lambda + y^\lambda - 1, 0))^{\frac{1}{\lambda}} & \text{otherwise.} \end{cases}$$

  where $T_D(x,y) = \begin{cases} 0, & \text{if } x,y \in [0,1), \\ \wedge(x,y) & \text{otherwise.} \end{cases}$.

  We have taken the value of $\lambda = -5$ given in [22] as best result.

- Structured Forests [16]. In this approach the authors propose an edge detector learned from information of local image patches, using Random Decision Forests. Originally the method works with *RGB* color images, so we have trained the model for computing edges in grayscale with the configuration parameters exposed in [16]. We denote this approach by *SF*.

- The two edge detector introduced in the next Section, *Case 1* (*C1*) and *Case 2* (*C2*), built from ODM functions.

Anew for all methods, to thin the feature image, ($S3$), we compute prior orientations and subsequently NMS process, both by Kovesis' functions [27]. We finish binarizing the thinned image, ($S4$), [15, 36].

7.0.3. Comparison method

It is well-known that the procedure to evaluate the performance of an edge detector is an open problem [29]. In this paper, to make the comparison, ($S5$), we have applied the approach given by Martin *et al.* [34]. This approximation considers that we are dealing with a binary classification problem (each pixel is classified as an element of the edge or is not classified as an element of the edge) and compares the output image given by an edge detector method with the ones generated by humans (ground truth images). To do so, a confusion matrix is performed.

Thereupon, bearing in mind all previous indications, for each image in BSDS500, the measures $PREC$, REC and $F_{0.5}$ of a given edge detection method are those produced in the comparison with the ground

Method	$PREC$	REC	$F_{0.5}$
C1	0.579	**0.794**	0.653
C2	0.602	0.765	0.654
FM_{SS}	0.572	0.719	0.615
C	0.687	0.618	0.631
G_{S_P}	0.649	0.649	0.650
G_{S_M}	0.661	0.665	0.641
SF	**0.753**	0.645	**0.682**

Table 1: Comparison of ODM functions approach with other edge detection methods as Gravitational, Fuzzy Morphology, Structured Forest and Canny in terms of $PREC$, REC and $F_{0.5}$.

truth images for which $F_{0.5}$ is maximal. As we have previously stated, there exist a number of ground truth images for each original image, so in our experiments (see also [30]) we compare the solution given by any edge detection method with all of the ground truth images. Then, the triplet $(PREC, REC, F_{0.5})$ having the greatest $F_{0.5}$ is considered as the evaluation of the detection method for that image, i.e., for us, the highest $F_{0.5}$ value means that the solution obtained by the edge detection method is the closest to a human, so, in this sense, is the best solution.

Moreover, we have also used a one-to-one pixel matching algorithm to map the edge pixels in the output image and the ground truth ones. This matching allows some spatial tolerance (in our case, as much as 2.5% of the diagonal of the image), so that an edge pixel can be slightly displaced from its true position, yet being considered as correctly classified.

In order to do the pixel-to-pixel matching, we use the method presented by Estrada and Jepson [19] (its implementation can be found in [26]).

8. Experimental results

In this section we provide the results obtained by our new proposal, as an individual feature extractor and as a consensus feature image, combined with other feature extraction techniques.

8.1. Case 1 and Case 2 vs. the other methods

In Table 1 the results of each edge detection method are indicated displaying the average of $PREC$, REC and $F_{0.5}$ for the 200 test set images. In terms of REC we can deduce that we have obtained better results than the Canny method with all of our two methods (*Case 1*, *Case 2*), i.e., not including a lot of false positives. On the other hand we may observe that FM_{SS} combines a medium precision with a very high recall, therefore the majority of edges are detected at the cost of including a high number of false positives. In general, the best results are when $PREC$ and REC have similar values, that is, considering $F_{0.5}$ as an overall quality measure. In this case, the results achieved with *Case 1* and *Case 2* are competitive with the ones obtained with Canny and gravitational forces. Particularly, we obtain better results than the ones with Canny and similar to those using the gravitational edge detector. If we focus on the results obtained by the Structured Forest method, it obtains the highest values in terms of $F_{0.5}$, this is due to the high value of $PREC$ as it detects more edges than the rest of the methods although having a lowest REC. So, the results obtained with Case 1 and with Case 2 are better than those obtained with Canny, Fuzzy Morphology, Gravitational with the maximum and Gravitational with the probabilistic sum, but worse than those obtained with SF.

Next, in Table 2 we show the number of images in the dataset for which it is the best or worst performer (in terms of $F_{0.5}$).

We observe that *C1* and *C2* have the lowest values in terms of worst images. Moreover, the number of images where we are the best result outperforms the ones obtained by all the other methods except Structured Forest, which remains the best performer.

	Method											
	*		FM_{SS}		C		G_{S_P}		G_{S_M}		SF	
	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗
C1	44	17	14	86	6	39	29	**16**	21	19	**86**	23
C2	50	**9**	16	89	6	39	27	20	16	16	**86**	27

Table 2: Comparison of best (✓) and worst (✗) approaches for 200 images of (BSDS500) in terms of $F_{0.5}$.

8.2. Consensus edge detector vs. the other methods

In this subsection we are going to build combinations without repetition of the feature images obtained with the seven edge detection methods that we have considered in this work (Case 1, Case 2, Canny, Gravitational with the probabilistic sum, Gravitational with the maximum, Fuzzy Morphology and SF). Specifically, we are going to consider combinations of two (21 combinations), three (35 combinations), four (35 combinations) and five (21 combinations) feature images. For each of the considered combinations, we calculate the value of the measures $PREC$, REC and $F_{0.5}$ (Tables 4- 7).

8.2.1. Combinations of two feature images

First, in Table 4 we consider combinations of two feature images obtained with two different edge detectors. Since in this work we consider 7 edge detectors, we have 21 possible combinations, which appear in the first column of the Table. For each of these combinations, we calculate the values of the PREC, REC and $F_{0.5}$ measures. we observe that the combination of Case 1 with SF provides the best result (in terms of the measure $F_{0.5}$). Moreover, the $F_{0.5}$ value that we get for this combination of Case 1 and SF is better than the $F_{0.5}$ value obtained for SF individually (see Table 1).

8.2.2. Combinations of three feature images

Next, we consider combinations of three feature images obtained with three different edge detectors. The 35 possible combinations appear in the first column of Table 5 and for each of them we make the same study for the PREC, REC and $F_{0.5}$ values. In this case, we observe that the best result in terms of the measure $F_{0.5}$ is the combination of Case 1 with SF and the detector based on fuzzy morphology. But the $F_{0.5}$ value that we obtain, although better that the corresponding one for SF in Table 1, is worse than the one for the combination of Case 1 and SF considered in Table 4.

8.2.3. Combinations of more than three feature images

Finally we study combinations of four (Table 6) and five (Table 7) feature images. In both cases, we see that the best result according to $F_{0.5}$ is obtained from the combination of our cases, SF and some of the other methods, but the resulting $F_{0.5}$ value, although bigger than the one of SF in Table 1, does not overcome neither the combination of Case 1 with SF nor the combination of Case 1 with SF and the detector based on fuzzy morphology. Even more, the best combination in Table 7 gets a worst result (according to $F_{0.5}$) than the best combination in Table 6.

8.3. Experimental results in terms of the best and worst performance

In Tables VII-X we compare the different combinations of methods with the individual methods. We study for how many images from the set of 200 each method provides the best and the worst results. The first column contains the different combinations of methods. In the row corresponding to each of these methods the 200 studied images are considered. In the column marked with * it is displayed in how many of the 200 images the method to which the row corresponds provides the best (tick) and the worst (x) results. In the other columns it is displayed the number of images among the 200 in which the method at the heading of the column provides the best and the worst results.

Table 3: Complexity comparison of the $S2$ phase of each of the alternatives used in the experiments. We consider N the number of pixels, p the number of features and n_{trees} the number of trees.

Approach	Training	Extraction
Canny		$O(NlogN)$
Gravitational		$O(N)$
Fuzzy Morphology		$O(N)$
Structured Forest	$O(N^2pn_{trees})$	$O(pn_{trees})$
ODM-functions ($C1$, $C2$)	$O(N^2)$	$O(N)$

8.3.1. Combinations of two feature images

In Table VII we observe that the combination of Case 1 with SF and the combination of Case 2 with SF provide extremely competitive results, much better than those obtained with each method individually, including SF.

8.3.2. Combinations of three feature images

Regarding the number of images in which the best and the worst results are obtained by combinations of three methods, the result is even more relevant in Table VIII than in VII, since the combination of Cases 1 and 2 with SF provides the best result in 94 images and the worst in just 1.

8.3.3. Combinations of more than three feature images

In Tables IX and X we see that the best results are obtained by combination in which Cases 1 and 2 always appear, and moreover, such results are always very competitive. Furthermore, in all the cases the combination of methods provides the worst result in a very small number of cases.

8.4. Note about the time complexity

As we followed the Bezdek Breakdown Structure all the phases are common in the experiments, except for the feature extraction ($S2$), and hence is the complexity. To measure complexity we consider N the number of pixels in an image. Phase $S1$, as done by a convolutional approach can be done in $O(NlogN)$. Phases $S3$ and $S4$ are a post-processing over the feature image that can be done in $O(N)$. For phase $S2$ we have different time complexities as we can see in Table 3. As it can be observed for our approach and Structured Forest there is an additional complexity for the training step, but this extra phase is done once.

When analysing our consensus edge detector, as we only use the arithmetic mean to fuse the different feature images, depending on the number of images we have a complexity of $O(kN)$, being k the number of images.

8.5. Conclusions from the experimental study

From the experimental study that we have done, we arrive at the following conclusions:

- The combination of methods improves the results obtained by each method separately.
- As more methods are combined, the results for the best combination (according to the $F_{0.5}$ measure) worsen.
- The best results correspond always to combinations in which Case1 or Case 2 (or both) and SF appear.

As an overall conclusion we remark that when compared individually Case1, Case 2 and SF methods shed good results and using penalty functions for the consensus method make the results overcome the ones obtained with each one of the methods compared individually. This behaviour is due to the power of penalty functions that permit to choose the best element of each one of the feature image alternative.

Method	$PREC$	REC	$F_{0.5}$
C1-C2	0.586	**0.785**	0.654
C1-Canny	0.661	0.682	0.652
C1-FM_{SS}	0.582	0.784	0.651
C1-G_{S_P}	0.620	0.749	0.660
C1-G_{S_M}	0.625	0.731	0.654
C1-SF	0.715	0.724	**0.705**
C2-Canny	0.668	0.669	0.650
C2-FM_{SS}	0.602	0.754	0.649
C2-G_{S_P}	0.628	0.731	0.657
C2-G_{S_M}	0.635	0.715	0.653
C2-SF	0.720	0.710	0.701
Canny-FM_{SS}	0.675	0.650	0.644
Canny-G_{S_P}	0.677	0.666	0.651
Canny-G_{S_M}	0.674	0.648	0.641
Canny-SF	**0.728**	0.671	0.683
FM_{SS}*-*G_{S_P}	0.624	0.734	0.655
FM_{SS}*-*G_{S_M}	0.631	0.711	0.649
FM_{SS}*-SF*	0.722	0.708	0.701
G_{S_P}*-*G_{S_M}	0.654	0.687	0.650
G_{S_P}*-SF*	0.722	0.688	0.687
G_{S_M}*-SF*	0.725	0.673	0.681

Table 4: Comparison of penalty functions combining two feature images in terms of $PREC$, REC and $F_{0.5}$.

These results justify the introduction and the study of Case 1 and Case 2. So we propose to use always combinations of two methods. We leave for a future work the analysis of the causes of this improvement in the performance when different methods are combined.

9. Conclusions and Future Research

As a conclusion for this paper, we can make the following remarks. The goal of evaluating the validity of ODM functions, used to measure the changes of intensity between a pixel and its neighbours taking into account the direction defined by the vector obtained by ordering in a decreasing way such intensity changes, has been achieved, so such functions can be used to determine the existence of an edge. Furthermore, using different ODM functions in phase (S2) and bearing in mind $F_{0.5}$ measure,we can conclude that, among the considered cases, Case 1 and Case 2 individually are only surpassed by SF. This justifies the consideration of the new method to build ODM functions and the introduction of both Case 1 and Case 2.

Furthermore, if we consider combinations of methods, the combinations of Case 1, Case 2 or both with SF overcome the results obtained by any of the methods individually. But the more combined methods, the worse the results are as the consensus approach tries to be the closest to every alternative to be combined. So we consider that the best option is to combine Case 1 or Case 2 with SF.

With respect to future research lines, we consider that it is necessary to make a study for optimizing the parameters in Eq. 1, both in a general way and with specific types of images.

As we have already mentioned in the introduction, our proposal is outperformed by deep learning methods. But we would like to insist once more on the simplicity and interpretability of our method, in particular when the feature image is built using the expression in Eq. (1). Note that, opposed to deep neural networks, no training is needed in our case, and furthermore, the computational cost of our algorithms is very low compared to that of deep learning procedures. Nevertheless, in future works we intend to study the possible combination of our algorithms and those based in deep learning techniques.

Method	$PREC$	REC	$F_{0.5}$
C1-C2-Canny	0.648	0.704	0.656
C1-C2-FM_{SS}	0.591	**0.779**	0.654
C1-C2-G_{S_P}	0.614	0.758	0.661
C1-C2-G_{S_M}	0.620	0.741	0.656
C1-C2-SF	0.699	0.738	0.702
C1-Canny-FM_{SS}	0.654	0.693	0.655
C1-Canny-G_{S_P}	0.656	0.701	0.659
C1-Canny-G_{S_M}	0.656	0.686	0.652
C1-Canny-SF	0.710	0.700	0.689
C1-FM_{SS}*-*G_{S_P}	0.615	0.757	0.660
C1-FM_{SS}*-*G_{S_M}	0.620	0.737	0.654
C1-FM_{SS}*-SF*	0.701	0.737	**0.704**
C1-G_{S_P}*-*G_{S_M}	0.638	0.723	0.658
C1-G_{S_P}*-SF*	0.695	0.725	0.693
C1-G_{S_M}*-SF*	0.701	0.706	0.687
C2-Canny-FM_{SS}	0.660	0.683	0.653
C2-Canny-G_{S_P}	0.660	0.692	0.656
C2-Canny-G_{S_M}	0.664	0.674	0.649
C2-Canny-SF	0.713	0.692	0.687
C2-FM_{SS}*-*G_{S_P}	0.621	0.746	0.659
C2-FM_{SS}*-*G_{S_M}	0.626	0.728	0.653
C2-FM_{SS}*-SF*	0.704	0.726	0.700
C2-G_{S_P}*-*G_{S_M}	0.643	0.713	0.656
C2-G_{S_P}*-SF*	0.700	0.717	0.691
C2-G_{S_M}*-SF*	0.707	0.700	0.686
Canny-FM_{SS}*-*G_{S_P}	0.664	0.689	0.657
Canny-FM_{SS}*-*G_{S_M}	0.665	0.670	0.648
Canny-FM_{SS}*-SF*	**0.719**	0.691	0.689
Canny-G_{S_P}*-*G_{S_M}	0.673	0.668	0.650
Canny-G_{S_P}*-SF*	0.714	0.680	0.679
Canny-G_{S_M}*-SF*	**0.719**	0.663	0.672
FM_{SS}*-*G_{S_P}*-*G_{S_M}	0.640	0.714	0.655
FM_{SS}*-*G_{S_P}*-SF*	0.704	0.716	0.693
FM_{SS}*-*G_{S_M}*-SF*	0.712	0.696	0.688
G_{S_P}*-*G_{S_M}*-SF*	0.711	0.682	0.678

Table 5: Comparison of penalty functions combining three feature images in terms of $PREC$, REC and $F_{0.5}$.

Method	$PREC$	REC	$F_{0.5}$
C1-C2-Canny-FM_{SS}	0.644	0.711	0.657
C1-C2-Canny-G_{S_P}	0.647	0.719	0.662
C1-C2-Canny-G_{S_M}	0.650	0.699	0.654
C1-C2-Canny-SF	0.702	0.709	0.690
C1-C2-FM_{SS}*-*G_{S_P}	0.611	**0.763**	0.661
C1-C2-FM_{SS}*-*G_{S_M}	0.617	0.744	0.655
C1-C2-FM_{SS}*-SF*	0.694	0.739	**0.700**
C1-C2-G_{S_P}*-*G_{S_M}	0.629	0.738	0.660
C1-C2-G_{S_P}*-SF*	0.684	0.738	0.693
C1-C2-G_{S_M}*-SF*	0.685	0.724	0.688
C1-Canny-FM_{SS}*-*G_{S_P}	0.649	0.714	0.661
C1-Canny-FM_{SS}*-*G_{S_M}	0.651	0.698	0.654
C1-Canny-FM_{SS}*-SF*	0.705	0.709	0.691
C1-Canny-G_{S_P}*-*G_{S_M}	0.656	0.698	0.657
C1-Canny-G_{S_P}*-SF*	0.697	0.707	0.685
C1-Canny-G_{S_M}*-SF*	0.704	0.687	0.678
C1-FM_{SS}*-*G_{S_P}*-*G_{S_M}	0.629	0.737	0.659
C1-FM_{SS}*-*G_{S_P}*-SF*	0.683	0.738	0.694
C1-FM_{SS}*-*G_{S_M}*-SF*	0.688	0.721	0.687
C1-G_{S_P}*-*G_{S_M}*-SF*	0.688	0.715	0.683
C2-Canny-FM_{SS}*-*G_{S_P}	0.654	0.706	0.660
C2-Canny-FM_{SS}*-*G_{S_M}	0.655	0.688	0.652
C2-Canny-FM_{SS}*-SF*	0.707	0.703	0.689
C2-Canny-G_{S_P}*-*G_{S_M}	0.662	0.689	0.655
C2-Canny-G_{S_P}*-SF*	0.703	0.698	0.683
C2-Canny-G_{S_M}*-SF*	**0.709**	0.680	0.676
C2-FM_{SS}*-*G_{S_P}*-*G_{S_M}	0.631	0.731	0.658
C2-FM_{SS}*-*G_{S_P}*-SF*	0.690	0.729	0.693
C2-FM_{SS}*-*G_{S_M}*-SF*	0.697	0.711	0.687
C2-G_{S_P}*-*G_{S_M}*-SF*	0.693	0.708	0.682
Canny-FM_{SS}*-*G_{S_P}*-*G_{S_M}	0.662	0.687	0.654
Canny-FM_{SS}*-*G_{S_P}*-SF*	0.707	0.696	0.685
Canny-FM_{SS}*-*G_{S_M}*-SF*	**0.709**	0.678	0.676
Canny-G_{S_P}*-*G_{S_M}*-SF*	0.705	0.678	0.673
FM_{SS}*-*G_{S_P}*-*G_{S_M}*-SF*	0.696	0.707	0.683

Table 6: Comparison of penalty functions combining four feature images in terms of $PREC$, REC and $F_{0.5}$.

Method	$PREC$	REC	$F_{0.5}$
C1-C2-Canny-FM_{SS}*-*G_{S_P}	0.642	0.726	0.663
C1-C2-Canny-FM_{SS}*-*G_{S_M}	0.645	0.706	0.655
C1-C2-Canny-FM_{SS}*-SF*	0.695	0.715	0.689
C1-C2-Canny-G_{S_P}*-*G_{S_M}	0.651	0.707	0.658
C1-C2-Canny-G_{S_P}*-SF*	0.688	0.718	0.685
C1-C2-Canny-G_{S_M}*-SF*	0.690	0.702	0.678
C1-C2-FM_{SS}*-*G_{S_P}*-*G_{S_M}	0.625	0.742	0.660
C1-C2-FM_{SS}*-*G_{S_P}*-SF*	0.675	**0.744**	**0.692**
C1-C2-FM_{SS}*-*G_{S_M}*-SF*	0.678	0.733	0.687
C1-C2-G_{S_P}*-*G_{S_M}*-SF*	0.676	0.724	0.682
C1-Canny-FM_{SS}*-*G_{S_P}*-*G_{S_M}	0.650	0.709	0.659
C1-Canny-FM_{SS}*-*G_{S_P}*-SF*	0.689	0.718	0.686
C1-Canny-FM_{SS}*-*G_{S_M}*-SF*	0.694	0.699	0.679
C1-Canny-G_{S_P}*-*G_{S_M}*-SF*	0.692	0.698	0.676
C1-FM_{SS}*-*G_{S_P}*-*G_{S_M}*-SF*	0.679	0.726	0.685
C2-Canny-FM_{SS}*-*G_{S_P}*-*G_{S_M}	0.654	0.701	0.658
C2-Canny-FM_{SS}*-*G_{S_P}*-SF*	0.694	0.710	0.685
C2-Canny-FM_{SS}*-*G_{S_M}*-SF*	0.699	0.692	0.678
C2-Canny-G_{S_P}*-*G_{S_M}*-SF*	0.697	0.691	0.675
C2-FM_{SS}*-*G_{S_P}*-*G_{S_M}*-SF*	0.685	0.717	0.684
Canny-FM_{SS}*-*G_{S_P}*-*G_{S_M}*-SF*	**0.700**	0.691	0.677

Table 7: Comparison of penalty functions combining five feature images in terms of $PREC$, REC and $F_{0.5}$.

Acknowledgements

This work was supported by the the Slovak Research and Development Agency through grant APVV-18-0052 and the Grant Agency of the Czech Republic, through grant GACR 18-06915S.

References

[1] G. Beliakov, A. Pradera and T. Calvo, *Aggregation Functions: A Guide for Practitioners*, ser. Studies In Fuzziness and Soft Computing, Berlin, Germany: Springer-Verlag, 2007.

[2] G. Beliakov, H. Bustince, T. Calvo, *A practical Guide to Averaging Functions*, Springer, Heidelberg, 2016.

[3] H. Benninghoff, H. Garcke, "Image Segmentation Using Parametric Contours With Free Endpoints", *IEEE Transactions on Image Processing*, vol. 25, no. 4, pp. 1639–1648, 2016.

[4] J. Bezdek, R. Chandrasekhar, Y. Attikouzel, "A geometric approach to edge detection," *IEEE Transactions on Fuzzy Systems*, vol. 6, no. 1, pp. 52–75, 1998.

[5] H. Bustince, E. Barrenechea, T. Calvo, S. James, G. Beliakov, "Consensus in multi-expert decision making problems using penalty functions defined over a Cartesian product of lattices," *Information Fusion*, vol. 17, pp. 56–64, 2014.

[6] H. Bustince, E. Barrenechea, M. Pagola, J. Fernandez , "Interval-valued fuzzy sets constructed from matrices: Application to edge detection," *Fuzzy Sets and Systems* , vol. 160, no. 13, pp. 1819–1840, 2009.

[7] H. Bustince, E. Barrenechea, M. Sesma-Sara, J. Lafuente, R. Mesiar, A. Kolesárová, "Ordered directionally monotone functions. Justification and application," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 4, pp. 2237–2250, 2018.

[8] H. Bustince, G. Beliakov, G. Pereira Dimuro, B. Bedregal, R. Mesiar , "On the definition of penalty functions in data aggregation," *Fuzzy Sets and Systems*, vol. 323, no. 15, pp. 1–18, 2017.

[9] H. Bustince, E. Barrenechea, T. Calvo, S. James, G. Beliakov , "Consensus in multi-expert decision making problems using penalty functions defined over a Cartesian product of lattices," *Information Fusion*, vol. 17, pp. 56–64, 2014.

[10] H. Bustince, M. Pagola, E. Barrenechea, J. Fernandez, P. Melo-Pinto, P. Couto, H. Tizhoosh, and J. Montero, "Ignorance functions. an application to the calculation of the threshold in prostate ultrasound images," *Fuzzy Sets and Systems*, vol. 161, no. 1, pp. 20 – 36, 2010

[11] H. Bustince, J. Fernandez, A. Kolesárová, R. Mesiar , "Directional monotonicity of fusion functions," *European Journal of Operational Research*, vol. 244, no. 1, pp. 300–308, 2015.

	Method											
	*		FM_{SS}		C		G_{S_P}		G_{S_M}		SF	
	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗
C1-C2	42	**11**	16	88	6	39	31	20	18	16	**87**	26
C1-Canny	18	**2**	26	94	6	39	41	21	18	16	**91**	28
C1-FM_{SS}	43	**15**	11	87	6	39	34	19	20	16	**86**	24
C1-G_{S_P}	43	**6**	22	93	7	39	18	18	19	16	**91**	28
C1-G_{S_M}	30	**4**	22	93	6	39	33	21	17	15	**92**	28
C1-SF	**97**	**1**	20	96	4	39	26	21	15	16	38	27
C2-Canny	15	**1**	28	95	6	39	42	21	18	16	**91**	28
C2-FM_{SS}	38	**11**	14	88	6	39	37	19	17	15	**88**	28
C2-G_{S_P}	38	**3**	25	95	7	39	22	19	18	16	**90**	28
C2-G_{S_M}	26	**2**	24	94	6	39	36	21	15	16	**93**	28
C2-SF	**96**	**1**	20	96	4	39	29	21	15	16	36	27
Canny-FM_{SS}	16	**1**	29	96	3	38	42	21	20	16	**90**	28
Canny-G_{S_P}	15	**3**	28	96	7	38	37	20	19	15	**94**	28
Canny-G_{S_M}	9	**5**	28	96	7	37	43	21	19	13	**94**	28
Canny-SF	52	**2**	27	95	5	39	36	20	16	16	**64**	28
FM_{SS}*-*G_{S_P}	42	**9**	23	91	6	38	19	18	19	16	**91**	28
FM_{SS}*-*G_{S_M}	24	**4**	22	93	6	39	39	21	17	15	**92**	28
FM_{SS}*-SF*	**98**	**1**	20	96	4	39	28	21	16	16	34	27
G_{S_P}*-*G_{S_M}	18	**6**	26	95	7	38	37	19	17	14	**95**	28
G_{S_P}*-SF*	61	**6**	28	95	5	38	28	18	16	16	**62**	27
G_{S_M}*-SF*	48	**3**	29	95	5	39	36	20	15	16	**67**	27

Table 8: Comparison of best (✓) and worst (✗) approaches in terms of $F_{0.5}$ considering penalty approaches with two feature images.

	Method											
	*		FM_{SS}		C		G_{S_P}		G_{S_M}		SF	
	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗
C1-C2-Canny	30	**1**	21	95	6	39	38	21	16	16	**89**	28
C1-C2-FM_{SS}	41	**10**	14	90	6	39	33	19	18	16	**88**	26
C1-C2-G_{S_P}	44	**2**	22	95	6	39	22	20	17	16	**89**	28
C1-C2-G_{S_M}	34	**4**	22	92	6	39	34	21	17	16	**87**	28
C1-C2-SF	**93**	**2**	19	96	5	39	25	21	11	16	47	26
C1-Canny-FM_{SS}	22	**1**	24	95	6	39	40	21	17	16	**91**	28
C1-Canny-G_{S_P}	28	**3**	27	95	6	38	30	20	17	16	**92**	28
C1-Canny-G_{S_M}	22	**1**	26	96	6	39	37	21	17	15	**92**	28
C1-Canny-SF	**63**	**1**	26	95	5	39	30	21	13	16	63	28
C1-FM_{SS}*-*G_{S_P}	46	**11**	22	90	7	38	19	18	20	16	**86**	27
C1-FM_{SS}*-*G_{S_M}	33	**6**	20	92	6	39	32	21	18	14	**91**	28
C1-FM_{SS}*-SF*	**95**	**1**	19	96	3	39	23	21	13	16	47	27
C1-G_{S_P}*-*G_{S_M}	34	**3**	25	95	7	38	25	21	17	15	**92**	28
C1-G_{S_P}*-SF*	**85**	**0**	24	96	5	39	13	21	15	16	58	28
C1-G_{S_M}*-SF*	**67**	**1**	22	95	6	39	29	21	11	16	65	28
C2-Canny-FM_{SS}	20	**1**	25	95	5	39	41	21	18	16	**91**	28
C2-Canny-G_{S_P}	21	**2**	27	95	6	38	31	21	20	16	**95**	28
C2-Canny-G_{S_M}	15	**1**	28	95	7	39	41	21	18	16	**91**	28
C2-Canny-SF	52	**3**	27	94	6	39	36	20	16	16	**63**	28
C2-FM_{SS}*-*G_{S_P}	39	**6**	23	93	7	38	22	19	19	16	**90**	28
C2-FM_{SS}*-*G_{S_M}	25	**4**	21	94	6	39	38	21	18	14	**92**	28
C2-FM_{SS}*-SF*	**88**	**1**	16	96	3	39	29	21	15	16	49	27
C2-G_{S_P}*-*G_{S_M}	30	**3**	24	96	7	38	30	21	17	14	**92**	28
C2-G_{S_P}*-SF*	**81**	**0**	24	96	5	39	20	21	15	16	55	28
C2-G_{S_M}*-SF*	62	**1**	24	95	6	39	31	21	12	16	**65**	28
Canny-FM_{SS}*-*G_{S_P}	25	**2**	27	95	6	39	29	20	21	16	**92**	28
Canny-FM_{SS}*-*G_{S_M}	18	**3**	28	94	6	39	40	21	17	15	**91**	28
Canny-FM_{SS}*-SF*	**60**	**2**	26	95	5	39	33	21	16	16	60	27
Canny-G_{S_P}*-*G_{S_M}	16	**5**	27	94	7	38	39	20	17	15	**94**	28
Canny-G_{S_P}*-SF*	44	**3**	28	95	4	38	34	20	14	16	**76**	28
Canny-G_{S_M}*-SF*	37	**5**	28	93	4	39	39	20	12	16	**80**	27
FM_{SS}*-*G_{S_P}*-*G_{S_M}	28	**5**	25	95	7	38	30	21	16	13	**94**	28
FM_{SS}*-*G_{S_P}*-SF*	**85**	**2**	25	95	5	39	17	20	15	16	53	28
FM_{SS}*-*G_{S_M}*-SF*	62	**1**	23	95	5	39	34	21	13	16	**63**	28
G_{S_P}*-*G_{S_M}*-SF*	43	**2**	29	95	6	38	31	21	15	16	**76**	28

Table 9: Comparison of best (✓) and worst (✗) approaches in terms of $F_{0.5}$ considering penalty approaches with three feature images.

	Method											
	*		FM_{SS}		C		G_{S_P}		G_{S_M}		SF	
	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗
C1-C2-Canny-FM_{SS}	29	**0**	22	96	6	39	36	21	18	16	**89**	28
C1-C2-Canny-G_{S_P}	34	**1**	24	96	6	38	31	21	16	16	**89**	28
C1-C2-Canny-G_{S_M}	26	**2**	25	94	6	39	35	21	16	16	**92**	28
C1-C2-Canny-SF	**71**	**1**	23	95	4	39	28	21	13	16	61	28
C1-C2-FM_{SS}*-*G_{S_P}	46	**6**	19	93	7	38	22	19	18	16	**88**	28
C1-C2-FM_{SS}*-*G_{S_M}	36	**3**	18	94	6	39	34	21	17	15	**89**	28
C1-C2-FM_{SS}*-SF*	**97**	**3**	14	95	3	39	22	21	12	16	52	26
C1-C2-G_{S_P}*-*G_{S_M}	33	**2**	25	95	7	38	26	21	18	16	**91**	28
C1-C2-G_{S_P}*-SF*	**88**	**1**	20	96	5	39	11	21	16	16	60	27
C1-C2-G_{S_M}*-SF*	68	**0**	18	96	6	39	27	21	11	16	**70**	28
C1-Canny-FM_{SS}*-*G_{S_P}	35	**3**	23	95	7	38	27	20	17	16	**91**	28
C1-Canny-FM_{SS}*-*G_{S_M}	27	**1**	23	96	7	39	37	21	16	15	**90**	28
C1-Canny-FM_{SS}*-SF*	**70**	**1**	22	95	5	39	27	21	16	16	60	28
C1-Canny-G_{S_P}*-*G_{S_M}	24	**3**	25	95	7	38	32	20	19	16	**93**	28
C1-Canny-G_{S_P}*-SF*	65	**1**	25	95	5	39	22	21	15	16	**68**	28
C1-Canny-G_{S_M}*-SF*	47	**1**	25	95	6	39	35	21	14	16	**73**	28
C1-FM_{SS}*-*G_{S_P}*-*G_{S_M}	37	**2**	24	95	7	38	26	21	18	16	**88**	28
C1-FM_{SS}*-*G_{S_P}*-SF*	**86**	**1**	18	96	5	39	14	21	15	16	62	27
C1-FM_{SS}*-*G_{S_M}*-SF*	**70**	**1**	17	95	6	39	29	21	10	16	68	28
C1-G_{S_P}*-*G_{S_M}*-SF*	62	**0**	25	96	6	39	21	21	13	16	**73**	28
C2-Canny-FM_{SS}*-*G_{S_P}	24	**2**	25	95	6	38	33	21	19	16	**93**	28
C2-Canny-FM_{SS}*-*G_{S_M}	22	**2**	26	95	6	39	39	21	16	15	**91**	28
C2-Canny-FM_{SS}*-SF*	**68**	**2**	22	94	5	39	30	21	16	16	59	28
C2-Canny-G_{S_P}*-*G_{S_M}	22	**3**	25	95	7	38	35	20	17	16	**94**	28
C2-Canny-G_{S_P}*-SF*	63	**2**	27	95	5	38	20	21	14	16	**71**	28
C2-Canny-G_{S_M}*-SF*	40	**1**	26	95	5	39	34	21	14	16	**81**	28
C2-FM_{SS}*-*G_{S_P}*-*G_{S_M}	31	**3**	23	95	7	38	30	20	17	16	**92**	28
C2-FM_{SS}*-*G_{S_P}*-SF*	**87**	**1**	22	95	5	39	14	21	13	16	59	28
C2-FM_{SS}*-*G_{S_M}*-SF*	67	**1**	19	95	6	39	29	21	11	16	**68**	28
C2-G_{S_P}*-*G_{S_M}*-SF*	62	**1**	25	96	6	38	23	21	13	16	**71**	28
Canny-FM_{SS}*-*G_{S_P}*-*G_{S_M}	22	**3**	26	94	7	38	33	21	18	16	**94**	28
Canny-FM_{SS}*-*G_{S_P}*-SF*	65	**1**	25	96	4	38	24	21	13	16	**69**	28
Canny-FM_{SS}*-*G_{S_M}*-SF*	41	**3**	28	94	5	39	34	21	14	16	**78**	27
Canny-G_{S_P}*-*G_{S_M}*-SF*	37	**3**	29	95	4	38	34	20	16	16	**80**	28
FM_{SS}*-*G_{S_P}*-*G_{S_M}*-SF*	65	**2**	24	95	5	38	23	21	13	16	**70**	28

Table 10: Comparison of best (✓) and worst (✗) approaches in terms of $F_{0.5}$ considering penalty approaches with four feature images.

	Method											
	*		FM_{SS}		C		G_{S_P}		G_{S_M}		SF	
	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗
C1-C2-Canny-FM_{SS}*-*G_{S_P}	34	**2**	22	95	7	38	30	21	17	16	**90**	28
C1-C2-Canny-FM_{SS}*-*G_{S_M}	26	**0**	24	96	6	39	37	21	16	16	**91**	28
C1-C2-Canny-FM_{SS}*-SF*	**69**	**1**	21	95	5	39	29	21	13	16	63	28
C1-C2-Canny-G_{S_P}*-*G_{S_M}	28	**3**	25	94	7	38	31	21	17	16	**92**	28
C1-C2-Canny-G_{S_P}*-SF*	69	**1**	24	95	5	39	19	21	13	16	**70**	28
C1-C2-Canny-G_{S_M}*-SF*	48	**1**	24	95	5	39	32	21	15	16	**76**	28
C1-C2-FM_{SS}*-*G_{S_P}*-*G_{S_M}	36	**3**	22	94	7	38	28	21	17	16	**90**	28
C1-C2-FM_{SS}*-*G_{S_P}*-SF*	**86**	**1**	17	95	6	39	14	21	15	16	62	28
C1-C2-FM_{SS}*-*G_{S_M}*-SF*	**72**	**1**	16	95	6	39	26	21	11	16	69	28
C1-C2-G_{S_P}*-*G_{S_M}*-SF*	66	**1**	23	95	6	39	19	21	13	16	**73**	28
C1-Canny-FM_{SS}*-*G_{S_P}*-*G_{S_M}	25	**3**	25	94	7	38	31	21	19	16	**93**	28
C1-Canny-FM_{SS}*-*G_{S_P}*-SF*	**70**	**1**	23	95	5	39	22	21	15	16	65	28
C1-Canny-FM_{SS}*-*G_{S_M}*-SF*	53	**1**	24	95	6	39	32	21	12	16	**73**	28
C1-Canny-G_{S_P}*-*G_{S_M}*-SF*	52	**1**	24	95	5	39	27	21	12	16	**80**	28
C1-FM_{SS}*-*G_{S_P}*-*G_{S_M}*-SF*	66	**1**	20	95	6	39	22	21	14	16	**72**	28
C2-Canny-FM_{SS}*-*G_{S_P}*-*G_{S_M}	27	**4**	25	94	6	38	33	20	16	16	**93**	28
C2-Canny-FM_{SS}*-*G_{S_P}*-SF*	64	**2**	25	95	5	38	24	21	15	16	**67**	28
C2-Canny-FM_{SS}*-*G_{S_M}*-SF*	48	**1**	26	95	5	39	33	21	12	16	**76**	28
C2-Canny-G_{S_P}*-*G_{S_M}*-SF*	46	**1**	27	95	5	39	30	21	15	16	**77**	28
C2-FM_{SS}*-*G_{S_P}*-*G_{S_M}*-SF*	63	**2**	24	95	6	38	23	21	14	16	**70**	28
Canny-FM_{SS}*-*G_{S_P}*-*G_{S_M}*-SF*	48	**2**	26	95	5	38	28	21	15	16	**78**	28

Table 11: Comparison of best (✓) and worst (✗) approaches in terms of $F_{0.5}$ considering penalty approaches with five feature images.

[12] T. Calvo, A. Kolesárová, M. Komorníková and R. Mesiar, "Aggregation operators: properties, classes and construction methods," in *Aggregation Operators New Trends and Applications*, T. Calvo, G. Mayor and R. Mesiar, Eds. Physica-Verlag, Heidelberg, 2002, pp. 3-104.
[13] T. Calvo, G. Beliakov, "Aggregation functions based on penalties," *Fuzzy Sets and Systems*, vol. 161, no. 10, pp. 1420–1436, 2010.
[14] J. Canny, (1983). "Finding edges and lines in images," Tech. Rep., Massachusetts Institute of Technology, Cambridge, MA, USA, 1983.
[15] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679-698, 1986.
[16] P. Dollar and C. L. Zitnick, "Fast edge detection using structured forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 8, pp. 1558–1570, Aug 2015.
[17] A. Ferraz, C. Mallet, and N. Chehata, "Large-scale road detection in forested mountainous areas using airborne topographic lidar data," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 112, pp. 23 – 36, 2016.
[18] M.G. Forero, "Fuzzy thresholding and histogram analysis," in *Fuzzy Filters for Image Processing*, M. Nachtegael, D. Van der Weken, D. Van de Ville and E.E. Kerre, Eds. Springer, 2003, pp. 129–152.
[19] F. J. Estrada, A. D. Jepson, "Benchmarking Image Segmentation Algorithms," *International Journal of Computer Vision*, vol. 85, no. 2, pp. 167–181, 2009.
[20] M. Grabisch, J.-L. Marichal, R. Mesiar, E. Pap, *Aggregation Functions*, Cambridge University Press, Cambridge, 2009.
[21] M. Gonzalez-Hidalgo, S. Massanet, A. Mir, D. Ruiz-Aguilera, "On the choice of the pair conjunction-implication into the fuzzy morphological edge detector," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 4, pp. 872–884, 2015.
[22] M. Gonzalez-Hidalgo, A. Mir-Torres, D. Ruiz-Aguilera, J. Torrens, "On the generalization of the fuzzy morphological operators for edge detection," in *Proc. of the IFSA-EUSFLAT Conference*, 2015, pp. 1082–1089.
[23] M. Heath, S. Sarkar, T. Sanocki, K. Bowyer, "A robust visual method for assessing the relative performance of edge-detection algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 12, pp. 1338-1359, 1997.
[24] J. Juszczyk, E. Pietka, and B. Pycinski, "Granular computing in model based abdominal organs detection," *Computerized Medical Imaging and Graphics*, vol. 46, pp. 121 – 130, 2015, information Technologies in Biomedicine.
[25] A. Karaali, C. R. Jung, "Edge-Based Defocus Blur Estimation With Adaptive Scale Selection", *IEEE Transactions on Image Processing*, vol. 27, no. 3, pp. 1126–1137, 2017
[26] Kermit Research Unit (Ghent University), The Kermit Image Toolkit (KITT) [Online]. Available: www.kermitimagetoolkit.com
[27] P. D. Kovesi, (2012). "MATLAB and octave functions for computer vision and image processing," Centre for Exploration Targeting, School of Earth and Environment, Univ. Western Australia, Perth, Australia. Available: http://www.peterkovesi.com/matlabfns/.
[28] C. Lopez-Molina, H.Bustince, J. Fernandez, P. Couto, B. De Baets, "A gravitational approach to edge detection based on triangular norms," *Pattern Recognition*, vol. 43, no. 11, pp. 3730–3741, 2010.
[29] C. Lopez-Molina, B. De Baets, H. Bustince, "Quantitative error measures for edge detection," *Pattern Recognition*, vol. 46, no. 4, pp. 1125-1139, 2013.
[30] C. Lopez-Molina, B. De Baets, H. Bustince, "A frame work for edge detection based on relief functions," *Information Sciences*, vol. 278, no. 10, pp. 127–140, 2014.
[31] G. Lucca, J. Sanz, G. Pereira Dimuro, B. Bedregal, R. Mesiar, A. Kolesárová, H. Bustince, "Pre-aggregation functions: construction and an application," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 2, pp. 260–272, 2016.
[32] J. M. Malof, K. Bradbury, L. M. Collins, and R. G. Newell, "Automatic detection of solar photovoltaic arrays in high resolution aerial imagery," *Applied Energy*, vol. 183, pp. 229 – 240, 2016.
[33] D. Martin, C. Fowlkes, D. Tal, J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. of the 8th International Conference on Computer Vision*, vol. 2, 2001, pp. 416–423.
[34] D. Martin, C. Fowlkes, J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 530-549, 2004.
[35] R. Medina-Carnicer, F. Madrid-Cuevas, A. Carmona-Poyato, R. Muñoz-Salinas, "On candidates selection for hysteresis thresholds in edge detection," *Pattern Recognition*, vol. 42, no. 7, pp. 1284–1296, 2009.
[36] R. Medina-Carnicer, R. Muñoz-Salinas, E. Yeguas-Bolivar, L. Diaz-Mas, "A novel method to look for the hysteresis thresholds for the Canny edge detector," *Pattern Recognition*, vol. 44, no. 6, pp. 1201-1211, 2011.
[37] S. Ono, "L_0 gradient projection", *IEEE Transactions on Image Processing*, vol. 26, no. 4, pp. 1554–1564, 2017.
[38] J.M.S. Prewitt, "Object Enhancement and Extraction," *Picture Processing and Psychopictorics*, Academic Press, pp. 75-149, 1970.
[39] B. Schweizer, A. Sklar, "Associative functions and statistical triangle inequalities," *Pub. Math. Debrecen*, vol. 8, pp. 169–186, 1961.
[40] I. Sobel, G. Feldman, (1968). "A 3×3 isotropic gradient operator for image processing. Presented at a talk at the Stanford Artificial Project, 1968.
[41] V. Torre, T. Poggio, "On edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, pp. 147–163, 1984.
[42] T. Wilkin, G. Beliakov, "Weakly Monotonic Averaging Functions," *International Journal of Intelligent Systems*, vol. 30, no. 2, pp. 144–169, 2015.
[43] X. Wang, H. Ma, X. Chen, S. You, "Edge Preserving and Multi-Scale Contextual Neural Network for Salient Object

Detection", *IEEE Transactions on Image Processing*, vol. 27, no.1, pp. 121–134, 2018.
[44] S. Xie, Z. Tu, "Holistically nested edge detection", *International Journal of Computer Vision*, vol. 125, no. 1-3, pp. 3–18, 2017.
[45] Q. Zou, Z. Zhang, Q. Li, X. Qi, Q. Wang, S. Wang, "DeepCrack: Learning Hierarchical Convolutional Features for Crack Detection", *IEEE Transactions on Image Processing*, vol. 28, no.3, pp. 1498–1512, 2018.