On the Use of Quasi-arithmetic Means for the Generation of Edge Detection Blending Functions


Abstract—The edge detection process can be broken down into four basic transformations, modifying the image from the original presentation to the final edges one. The adoption of this framework makes the process far more understandable, and offers an starting point for the combination and comparison of different edge detection methods. In this work we analyze the role of the third of the transformations, the blending, where the edge features are combined to obtain the edginess values. This work studies the use of quasi-arithmetic means for the combination of the edge features. Moreover, we show results obtained with different operators on real images, in order to illustrate the importance of the blending phase in the edge detection process. Results will show the impact of the function selection in the final results.

I. INTRODUCTION

The process of edge detection in a image has always lacked a formal, widely accepted structure. Indeed, it is hard to find a definition of what an edge is, being Canny constraints [7] the most recognized approach. Some attempts have been carried out to characterize, at least, the different stages of the problem. An early mention to the problem was done by Torre and Poggio [29], stating that “the goal cannot be reached in a single step”. Bezdek et al. [5] introduced the first mathematical breakdown structure of the problem. It aimed to embrace all the previous work, as long as a wide variety of imaging data.

This work studies the role of the quasi-arithmetic means in the blending phase, where the edge features at every pixel are turned into edginess values. Moreover, we intend to point out the influence of this phase, sometimes ignored, in the final results.

The remainder of this article is organized as follows. In Section II the edge detection process, after Bezdek et al. [5], is introduced. Section III analyzes in depth the role of the blending phase in the literature. Some practical results using quasi-arithmetic means for the blending are shown in Section IV. To finish, some brief conclusions are depicted in Section V.

II. CHARACTERIZATION OF THE EDGE DETECTION PROCESS

After [5], the processing of an image in order to obtain the edges of its objects can be divided in four sequential phases, each of them represented by a function: conditioning (c), feature extraction (f), blending (b) and scaling (s). Figure 1 comprises the sequence. Considering an initial image E, the composition of all the functions produces the edges image G so that

\[ G = s(b(f(c(E)))) \] (1)

Each of the phases is characterized by the information it processes and its interpretation. Therefore, in order to compare or combine different edge detection methods, we only have to understand the meaning of the information at every step of the algorithm.

The first of the phases, conditioning, consists of the adequation of the image for edge detection. This might imply denoising, equalizing, smoothing or any other procedure ([3], [19], [21]). This function could even modify the number of values per pixel in the image, as it could include channel merging or dividing [26].

The second phase, feature extraction, is the most extensively covered in the literature. It consists of the extraction of information about the changes around each position of the image, i.e. to characterize how is the image changing at each position. Procedures used at this phase include,
among others, convolutions ([6], [11], [22], [25]), pattern matching ([5], [10]) and vectorial approaches ([14], [27]). This transformation from the intensity (or color) space to the features space is a recurrent recourse in image processing techniques, such as texture recognition and classification ([13], [20]).

In the blending phase the information about the features must be turned into a single value, usually denoted as edginess. In practical terms, we have the problem of combining a vector of features into a single value. Hence, the blending turns out to be an information aggregation problem.

Very often, the choice for blending is the magnitude of the vector, considering the features space is an Euclidean space ([7], [15], [27]). However, this is not necessarily the best option. We will show that the blending phase has remarkable effects in the results, and that its choice should be carefully meditated.

To finish, the last of the phases is the scaling, where the edginess value of each position in the image is turned into the desired presentation format. After Canny constraints [7], the presentation method typically consists of thin, binary edges. However, some other representations, as a Fuzzy Sets ([6], [23], [24]), could be used for different purposes.

This work focuses on the blending phase, where the features vector has to be combined into a single value.

III. THE BLENDING PHASE
A. The edge features

The procedures at the feature extraction phase generate a variable number of features per pixel. We will refer to this number as \( p \). Each feature might represent different kind of information, and their combination conforms the whole knowledge we have about the edges of the objects at each position of the image.

In the literature, techniques can be found generating a wide variety of features. In practical terms, this means that the blending techniques deal with feature vectors with different lengths. For instance, when the feature extraction is based on local contrast [9] or tn- and sn-processing [6], a single value per pixel is generated. Other direction-based methods analyze different intensity change characteristics of each pixel (CFED [11] uses four different features to characterize the edges). Alternatively, approaches based on fuzzy pattern matching ([10], [23]) can create as many features as patterns.

The number of features per pixel is typically increased using multi-scale detection ([12], [30]). This strategy consists of performing the feature extraction sequentially at different scales (i.e. considering different neighborhood sizes around a pixel), then using all of the generated features.

Despite it is not a rule, most of the popular edge detection methods assign every position in the image a directional estimation of the intensity changes in its neighborhood ([7], [14], [22], [25], [27]). This representation is usually understood as the \textit{gradient} of the image at a given point. Nevertheless, this consideration is not mathematically correct, as the image is not a continuous, differentiable surface [29]. The orientation of this gradient represents the direction the intensity is increasing toward, while its magnitude represents the strength of the change. All the practical examples in this work are considered for this kind of features.

B. Definition of blending functions

Given a vector of \( p \) features associated with each position in the image, Bezdek \textit{et al.} [5] define the blending functions as

\[
b : \mathbb{R}^p \to \mathbb{R}
\]

Bezdek \textit{et al.} [5] depict three different ways the blending functions can be generated in: Minkowski norms (illustrated in Figure 2), logistic functions (Figure 3) and trained models (based on the Takagi-Sugeno model [28]). Among them, the 1- and 2-powered Minkowski norms (Manhattan and Euclidean norms) are the most common ones in the literature.

After the original definition, these functions do not have to fulfill any special condition. This fact makes sense, since each of the features represents different information and the meaning of their values can vary.

However, if the features represent a gradient, some conditions could be established. One property to be expected is the symmetry with respect to all of the \( p \) axes. That is, negative feature values should have the same consideration than the positive ones, as the sign only represents directional information of the gradient. Moreover, \( b \) would rather be increasing on the absolute value of the each of the \( p \) components. To finish, as long as the features \((0, \ldots, 0)\) represent the absence of intensity changes, any blending function \( b \) should satisfy \( b(0, \ldots, 0) = 0 \).

In this way the \textit{blending} problem resembles to the \( p \)-ary aggregation one ([11], [4]), but for the range of the inputs. In order to overcome this problem we need to transform the features from \( \mathbb{R}^p \) to \([0, 1]^p\). Once we assume that negative and positive values should be treated in the same way, we only have to care about the normalization of the values. The normalization is carried out dividing every argument by the supreme absolute value that can be obtained with the feature extraction function \( f \). Once performed these transformations of the feature values, we can give rise to an alternative definition of the blending function:

\[
b : [0, 1]^p \to [0, 1]
\]

In order to illustrate the importance of the blending functions, we analyze the effects of taking different operators, all of them belonging to the same family. In this case, we work with quasi-arithmetic means. By using different functions, and seeing their impact in the final results, we will understand better the role of this phase of the edge detection process.

C. Quasi-arithmetic means as blending functions

Given a continuous, strictly increasing function \( g : [0, 1] \to [\infty, \infty] \), we define a quasi-arithmetic mean (also known as \( g \)-mean) \( M_g : [0, 1]^p \to [0, 1] \) as

\[
M_g(v) = g^{-1} \left( \frac{1}{p} \sum_{i=1}^{p} g(v_i) \right)
\] (2)
where \( \mathbf{v} = \{v_1, \ldots, v_p\}, \mathbf{v} \in [0,1]^p \).

We will limit our study to the features representing the components of a gradient. As we will work with grayscale images, the features vector associated with each pixel is a vector in the \( \mathbb{R}^2 \) space. Therefore, from now on, we will consider exclusively \( p = 2 \).

Specifically, we consider quasi-arithmetic means based on two kinds of generating functions \( g \):

- polynomials of the type \( g(x) = x^k \) (with \( k \neq 0 \))

- sigmoid functions.

In the first case, the polynomial-based blending functions \( b^{pol}_k \) are

\[
M_g(\mathbf{v}) = b^{pol}_k(\mathbf{v}) = \sqrt{\left(\frac{v_1^k + v_2^k}{2}\right)}
\]  

where \( \mathbf{v} \in [0,1]^2 \), \( \mathbf{v} = \{v_1, v_2\} \), represents the two-dimensional gradient and \( k > 0 \) is an arbitrary argument.

In this way, we are actually generating the family of H"{a}usser means [4]. It is also remarkable that, with \( k = 1,2 \), we would have the Manhattan and Euclidean norms, plotted in the Figure 2, up to a constant factor. Figure 4 includes examples of blending functions based on Eq. (3).

For the sigmoid-based means, namely \( b^{sig}_k \), we take as a basis a classic example of sigmoid function \( m : \mathbb{R} \to \mathbb{R} \), defined as

\[
m(x) = \frac{1}{1+e^{-k \cdot x}}
\]

where \( k > 0 \) determines the slope of the sigmoid. It will be used as an arbitrary argument, as it was for the polynomials generating functions.

In order to turn \( m(x) \) into a valid \( g \) function, as in Eq. (2), we only need to map the inputs from \([0,1]\) to \([-\infty, \infty]\). That is, given a mapping \( h : [0,1] \to [-\infty,\infty] \), to define \( g \) as

\[
g(x) = m(h(x)) = \frac{1}{1+e^{-k \cdot h(x)}}
\]

Having the function \( h(x) = ln(x) - ln(1-x) \), the \( g \) function is

\[
g(x) = \frac{1}{1+e^{-k \cdot (ln(x) - ln(1-x))}}
\]

considering the conventions \( ln 0 = -\infty, e^{-\infty} = 0, e^{\infty} = \infty \) and \( a - \infty = -\infty \) for all \( a \in \mathbb{R} \).

The blending functions generated after Eq. (2), using \( g(x) \) as in (6), will be denoted \( b^{sig}_k \), where \( k \) is the parameter in Eq. (4). Some examples of these quasi-arithmetic means are shown in Figure 5.

The functions plotted in Figures 4 and 5 give us an idea of the different functions to be obtained with each construction method. Furthermore, we can observe that the major differences between the functions happen when the difference between both components of the gradient is large. That is, close to \( \mathbf{v} = \{0,1\} \) and \( \mathbf{v} = \{1,0\} \). This is to be expected, since the means are idempotent. This implies that the mostly horizontal or mostly vertical edges (those producing gradients with one component significantly greater than the other) will be assigned very different edginess values, depending on the blending function.

IV. RESULTS

First, we have experimented with natural images, in order to visualize the variety of results when using different blending functions. We have selected four images, all of them included in the first column of the Figures 7 and 8. Images 1 and 2 have been extracted from the Berkeley Segmentation...
Dataset (BSDS) [16], while Images 3 and 4 are the well-known Lena and Cameraman images.

The algorithm for edge detection has been kept as simple as possible, making use of well-known, fully unsupervised techniques. Nevertheless, the main aim is to illustrate the impact of the blending functions, rather than finding optimal edges.

The processing steps are as follows:
1. **Conditioning**- Smooth the image with a \( \sigma = 1 \) Gaussian filter ([3], [29]).
2. **Feature extraction**- Filter the smooth image with Sobel operators [25], as displayed in Figure 6. The masks produce the horizontal \( (E_h) \) and vertical \( (E_v) \) components of the gradients, both in the range \([-1, 1]\). For instance, \( E_h = 0 \) means that there is no horizontal intensity change at all, while a \( E_h = -1 \) stands for a maximal intensity change, increasing to the left.

3. **Blending**- Combine the features generated in the previous phase. To do so, select \( b \) among those in Figures 4 and 5. The result of the combination is considered the edginess of the pixel, \( E_t = b(|E_h|, |E_v|) \).
4. **Scaling**- Binarize the edginess image. Use Non-Maximum Suppression (NMS) along with hysteresis, both of them introduced by Canny [7].
The NMS consists of analyzing and comparing the edginess of each pixel, in order to avoid the edge of an object to be thicker than one pixel. Even if the edges are thinned, the non-discarded pixels keep the edginess value.

Then, the hysteresis procedure discriminates which of the intensity changes are relevant enough to be selected as true edges, by means of a double thresholding. To determine the thresholds, we use the fully automatic method by Medina-Carnicer et al. [18].

The results of the edge detection are displayed in Figures 7 (blended images) and 8 (scaled images, satisfying the Canny constraints).

We recall that better results could have been achieved by using other procedures at steps (1), (2) and (4). However, maximizing the performance is not the goal of this work.

We can observe how the final results diverge when using different functions. The most significant cases are the mostly horizontal or vertical lines, as the silhouette of the building in the Image 1 or the background object in the leftmost part of the Image 3. These edges generate high edginess values with both $b_{0.25}^{pol}$ and $b_{1.00}^{pol}$. Hence, they are finally selected as edges. However, those operators also miss some edges. An example can be found on the top of the hat in the Image 3, having that $b_{0.25}^{pol}$ produces the whole silhouette, while $b_{0.25}^{pol}$ and $b_{0.00}^{pol}$ miss some segments. This is due to the fact that the edginess of these pixels are now relatively larger than before, compared to the rest of the pixels in the image. Therefore, they are likely to be selected as true edge points at the hysteresis procedure.

However, these are visual considerations. We have performed a second experiment in order to quantify the impact of the blending functions. To do so, we have replicated the edge detection process on images in the test set of the BSDS. This set contains 100 natural images, along with some hand-made binary edges. Using the consensus ground truth technique by Fernández-García et al. [8], we have created a single binary ground truth image for each test image. Then, we have used Baddeley error metrics (BEM) [2] to quantify the distance of the results of the detector with respect to the ideal solution.

Let $A$ and $B$ be two binary edge images of size $M \times N$, and let $S = \{1, \ldots, M\} \times \{1, \ldots, N\}$ be the set of their positions. Their $\alpha$-BEM, also $\Delta_\alpha^A(A, B)$, is defined as

$$\Delta_\alpha^A(A, B) = \left[ \frac{1}{|S|} \sum_{s \in S} |w(d(s, A)) - w(d(s, B))|^\alpha \right]^\frac{1}{\alpha} \tag{7}$$

where $d(s, A)$ represents the distance from the position $s$ to the closest edge point of the image $A$ and $w : \mathbb{R} \to \mathbb{R}$ is a concave, increasing function used for weighing.

We will use $\alpha = 2$, the euclidean distance $d$, and $w(x) = x$, as in [17].

The average performance (average BEM) of the algorithm, depending on the blending function is displayed in the Table I. The table also includes the percentage of images where each blending functions obtain the best and worst performance all over the 6 possible choices.

First, we see that the results are very similar, being the difference between the best one ($b_{0.25}^{pol}$) and the worst one ($b_{1.00}^{pol}$) around 2%. This contrasts with the visual results in the Figures 7 and 8, where results obtained with different functions vary significantly. Moreover, all of the methods produce a significant amount of best and worst results.

In general, we observe that the automatic selection of the blending functions is complicated, at least having no a priori information about the edges we are looking for. Even if using a relatively small set of functions, all of them belonging to the same family, the results obtained in the experiments happen to be very different. Hence, these functions should be selected regarding the expected results, and the characteristics of the edges to be empowered.

<table>
<thead>
<tr>
<th>Table I</th>
<th>Experimental results on the Berkeley Segmentation Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_{0.25}^{pol}$</td>
<td>34.07</td>
</tr>
<tr>
<td>$%$ Best results</td>
<td>21</td>
</tr>
<tr>
<td>$%$ Worst results</td>
<td>7</td>
</tr>
</tbody>
</table>

V. CONCLUSIONS

In this work we have analyzed the edge detection process breakdown by Bezdek et al.. We have then focused in the blending phase, where the edges features are combined, proposing the use of quasi-arithmetic means. We have illustrated how the choice of the blending function, even from a relatively small and homogeneous pool of candidates, has significant influence in the final results. This influence has been depicted visually, then quantified using the BEM.

In this case, given the nature of the function we have used (all of them means, hence idempotent), the differences have arisen in the horizontal and vertical edges. However, if considering other options the possibilities for edge features interpretation are infinite. This result might look like obvious, but in the literature the use of different operators for the blending is not deeply studied.
Fig. 7. Blended images using different blending functions
Fig. 8. Scaled images using different blending functions
ACKNOWLEDGMENT

This work has been partially supported by the Spanish Ministry of Education and Science (MEC), project TIN2007-65981, and by the Research Services of the Public University of Navarre.

REFERENCES