In real world images, many algorithms for adaptive contours detection exist and various improvements to the contours detection have been proposed. The reason for such diversity is that real world images contains heterogeneous mixtures of features and each of the available algorithms exploits some of these features. Thus, depending on the image, different algorithms shows different quality of result. In this paper we propose a method that improves the result adaptive contours detection by using an algorithm selection approach. Previous methods using the algorithm selection approach have been focusing only on images with a particular class of features (artificial, cellular) because of the complexity of real world images. In order to successfully solve this problem we first determine a set of distinctive features of each algorithm using machine learning. Then using these distinctive features we teach an algorithm selector to select best algorithm when a set of features is provided. Finally, we propose a method to split the input image into sub regions that are selected in such a manner that improves the quality of the image processing result. The proposed algorithm is verified on the set of benchmarks and its performance is comparable and better in many cases than the currently best contour detection algorithms.

KEYWORDS: contour detection in real-world images, algorithm selection, machine learning

1. Introduction

Image Segmentation is a well known task used in many areas of machine vision and image processing [23]. One of the main step of the segmentation process is the contour detection. Contour detection consists in assigning pixels to contours in such manner that when used for the segmentation, the resulting regions corresponds as close as possible to regions resulting from human segmentation: the contours are used to determine regions (segments) that in the ideal case correspond to symbolic and meaningful regions created by human segmentation.

The segmentation (and the contour detection) of natural images is a very difficult task because human is using low level (pixel level), mid level (region level) and high level (semantic level) [29] to segment an image. In general the computer algorithms use only the low level and mid level features (for example [6, 16, 24]). Consequently detecting symbolic areas and determining their contour only from the image information is very challenging because often contours are crossing, overlapping and often disappear and reappear with respect to the background.

Segmentation has also been studied as a process that is performed on selected regions of the image. The image is first analyzed and split according to some selected features and criteria and then the regions are segmented to obtain the best possible contour delimiting the segments. The general problem of these methods are that they are quite specific and in general cannot be used for automated segmentation on the same level as algorithms segmenting images as one.

In this paper we propose a method for selecting segmentation algorithms for natural images segmentation. For this purpose, we define what does it mean to have a successful selection and we use machine learning to determine the best algorithm on a case-by-case basis. The main contributions of this paper are as follow:

1. We show that using a machine learning approach to algorithms selection for real world image processing improves the general performance of the image processing when compared to individual algorithms.
2. The splitting of the images and the image processing on a regional basis improves the performance of the contour detection quality.
3. Despite the relatively bad results in machine learning, the resulting mechanism selection when applied to test images for contour detection still outperforms the individual algorithms applied to whole images.

This paper is organized as follows. Section 2 describes previous related work. Section 3 explains the particular details of the algorithm selection approach and Section 4 describes the technique called ‘Image Splitting’ that is used to minimize the overall processing requirements of the segmentation approach. Section 5 describes the machine learning of the algorithm selection and Section 6 describes the image reconstruction process. Finally Section 7 shows the experiments performed and Section 8 concludes this paper.
2. Previous Related Work

Due to the importance and complexity of the segmentation problem of natural images many algorithms have been proposed [1, 2, 6, 8, 9, 15, 16, 21, 24]. In general these algorithms are multiple level processes that use information extracted and built on various level of the image representation. However each of the segmentation algorithms has certain strong and weak points. This is because the natural images are highly heterogeneous and thus it is not possible to account for all possible variations when designing robust algorithms. Some of the used low level features are oriented gradients [17], brightness difference, color gradients, salience, etc. Such low level and in general local features are then modified using higher level strategies. Some of such strategies are the global probability of contour [15], ultrametric contour map [2], graph cut [6], total variation method [9], etc.

The image analysis and parsing has also been studied for the purpose of segmentation. For instance [12] studied images based on their color-space to determine the correct number of regions for segmentation. Gould [11] used regions based detection contrary to the traditional window sliding method: the image is decomposed into a set of semantically consistent regions and using an energy function combines all region into a semantical representation of the image. Another approach in splitting image in regions was proposed by [26] where an over segmentation was merged into regions and then into objects in a unified framework of segmentation and object recognition. Segmentation has also been studied as a process resulting from segmenting image by specific regions. In [1] the authors present a segmentation based on saliency in such manner that the salience is used as indicator where what areas are to be segmented.

The algorithm selection paradigm is quite old [22] and still today is not a main stream approach to general robotics or image processing. This is mostly due to the fact that selecting a candidate algorithm from a set of available algorithms is not a trivial task. For restricted image content this approach however was successful. For instance [27] used algorithm selection to decide which edge detector is best on a set of artificial images with various degrees of noise. In [25] the authors proposed to use the algorithm selection method on images of cells where the edges depended not only on the amount of the noise but also on the quality of the image. In a more restricted framework, several works have been studying the parameter selection of one segmentation method such as [14, 19, 21] however these works have not been using any kind of machine learning.

3. Algorithm Selection

The general idea behind the algorithm selection is to select a unique algorithm for a particular set of features extracted from the image. However, in general it is not possible to select effectively the best algorithm for natural images because

1. The images are heterogeneous — to exploit all possible combinations of features used to determine the best algorithm an infinitely large memory is required and all combinations of features are computationally not possible to estimate.
2. The algorithms use various features — vision processing algorithms in general exploit a particular set of features or their combinations as a starting point. Because of the natural heterogeneity of images, algorithms should be used on image sub regions corresponding to peak performance.
3. Different algorithms can have equivalent performance on a given image due to statistical averaging of the overall performance.

To effectively improve the image processing by the algorithm selection two problems must be solved in parallel. First, as was already previously introduced a robust mapping from a set of features to algorithms is required. Such mapping would allow to clearly separate the different algorithms based on their best results. The selection of algorithms for real world images containing natural and artificial elements such as forest, plants, animals, cars, buildings and so on, requires in the ideal case features that capture properties of the image that will allow to identify most appropriate algorithms. The main problem of selecting a multi-level algorithm from only low level features is difficult because the estimation of what algorithm should be used is done only based on partial (low level) features. Contrary to two approaches [25, 27] where the authors used simple algorithms such as Canny or the Prewitt edge detectors local features are more than adequate to select such algorithms, and thus the same task was simpler because the algorithms and the available features were of the same level.

Second, the quality of result of the algorithmic segmentation (contour detection) depends on both local and global features. This means that a real world image, being a mixture of various features, will have areas that are ideally suited for one particular algorithm and areas that different algorithm would segment with a better result. Thus, splitting the image into such regions that would maximize the quality of algorithms results given a image region will lead to an overall performance increase. The image splitting consists thus in separating the image into regions of certain properties and with certain mixture of features allowing to select the most fitting algorithm. As will be seen later, the size and the property of the regions require that it does not only degrade the performance of algorithms but that it provides the best possible result.
Definition 1 (Algorithm Selection). Given is a set of inputs $I = \{i_0, \ldots, i_k\}$ and a set of available algorithms $A = \{o_0, \ldots, o_l\}$ such that $A \times I = O$ with $O = \{o_i, \ldots, o_p\}$. Let $F^j = \{f_0, \ldots, f_n\}$ be a set of features that are obtained for each input $i_j$, a set of target outputs $O^j = \{o^j_0, \ldots, o^j_q\}$ corresponding to the desired outputs of the processing for each input in $I$ respectively and a measure of performance $\rho > 0$. For all $a_i \in A$ and input $i_m$, the performance is given as:

$$\rho(j, m) = \frac{1}{1 + |(o^j_m) - (o^m_j)|^2}$$

and the desired mapping is then $M : F \rightarrow A$.

For instance, let $o^m_j$ is the output of a detection algorithm represented as a black and white image where the white pixels indicate the detected color. Similarly, $o^j_m$ is the target, represented in a same manner as the $o^m_j$. To evaluate the quality of the detection algorithm, the pixel-by-pixel difference between the two results is computed as shown in Eq. (1). The resulting value in the [0,1] interval represents the matching of the algorithm output to the desired target. Equation (1) shows only a simple evaluation function and in general more complex evaluations are used such as the F-measure for instance (§3.1).

3.1 Algorithm evaluation

To learn the mapping $A$ a measure of the quality result is required. Equation (1) shows the usage of the pixel to pixel correspondence but in general such one-to-one correspondence is not well suited for artificial intelligence problem evaluation because it does not take into account the amount of false-positives and true-negatives results.

A common measure to evaluate a result of contour detection or segmentation is the F measure [18]. It is well known in the area of segmentation because it accounts well for both the amount of generated noise and for the precision of the regions/contours. F-measure takes into account both the false-positives as well as the true-negatives and thus offers a much higher precision of the evaluation. This means that using F-measure one calculates the ratio of result points that match the target points $|(o^j_m) \circ (o^j_m)|$ over the number of the target points (this is called precision) and the ratio of result points that match the target points over the result points $o^j_m$ (this ratio is called recall). This measure is more precise for the image processing evaluation.

Using this knowledge, the evaluation function used to determine the quality of the generated boundaries takes the form shown in Eq. (2). The first term in the left side of Eq. (2) represents the precision and the last term represents the recall.

$$\rho(j, m) = \frac{|(o^j_m) \circ (o^j_m)|}{o^j_m} + \frac{|(o^j_m) \circ (o^j_m)|}{o^j_m}.$$  

To evaluate the boundary results, the thresholding approach [17] is used. In the thresholding approach, the resulting gray scale image that contains the boundary, is iteratively thresholded to various threshold values and for each value of the threshold it is evaluated. Among all the obtained values of the F-measure, the best one is kept as the final score.

Table 1 shows comparative results of selected segmentation methods. The thresholding experiment was performed with 1, 5, 10, 50, and 100 points; this means that each image is tested for 1, 5, 10, 50, and 100 different thresholds of the intensity respectively. Observe that depending on the thresholding the quality of the resulting boundaries changes considerably.

<table>
<thead>
<tr>
<th>Threshold points</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.72</td>
</tr>
<tr>
<td>50</td>
<td>0.71</td>
</tr>
<tr>
<td>10</td>
<td>0.71</td>
</tr>
<tr>
<td>5</td>
<td>0.7</td>
</tr>
<tr>
<td>1</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Finally, note that the thresholding is not the best possible evaluation for boundary detection algorithms; it does not allow to determine the best algorithm in an automated manner. In general, for each input image and for each algorithm the best result is obtained using a different threshold. Thus for an arbitrary input image without a ground truth available it is not possible to determine the best threshold for the desired boundaries. Having a reliable selection mechanism can solve this problem by selecting such algorithm that will minimize the number of undesired pixels. This allows to determine a set of algorithms that are the most suited for real-time and real world unsupervised processing.

However as was shown in previous works [2, 15, 24], the F measure can favor such segmentations that do not result in meaningful segments. And thus it is even today difficult to objectively determine the best segmentation for a particular application.
4. Image Splitting

Natural images contain objects of various nature (artificial, natural) and of various properties (color gradient, textures, shading) that can be grouped together based on similarity of given features. Multi-level contour detection algorithms use information generated on various levels of processing: pixel level (color gradients, textures, etc.), region level graph partitions, geodesics or so. Natural images contain a non-homogeneous mixture of features: the quality of the result of each algorithm is proportional to the spatial distribution of features values in the image. Consequently, selecting the best algorithm for a particular image results in an algorithm that in average has the best quality of result over all of the areas in the image.

In order to improve the contour detection, a method of splitting images to such regions that maximize different algorithms quality of output is desired. There are several criterion that must be satisfied to allow this approach to be meaningful for the whole image segmentation:

1. The processing of an image by image regions must allow similar performance to the processing of the image as a whole. This a restriction in particular to the size of the image regions because too small regions will contain too much noise but too large regions will not allow regions with enough homogeneous feature vectors.

2. The image regions should contain regions of interest. Regions of image that do not contain regions of interests will in general provide resulting edges that are not part of the desired edges thus increasing the overall noise. This is a natural consequence of the fact that each algorithm for segmentation attempt to generate final contours using the most salient edges. Thus generating subregions of the image containing relevant edges will increase the chance of such edges to be detected while not relevant edges can be successfully eliminated.

3. The image regions should improve the processing time of the whole image.

Thus, for a given image with features $F_{av} = \{f_0, \ldots, f_j\}$ averaged over the $m \times n$ pixels (size of the image) a desired region is is given by

$$r_{i, j} = \{r_{i, j, x, y} \mid d_p(r_{i, j, x, y}) > a_q(r_{i, j, x, y}) \} \forall p, q$$

$$\frac{1}{m} \sum_{i=1}^{m} \rho(m, m_{best}) \geq \rho(r_{1,1,m,n,k})$$  \quad (3)

The first line of Eq. (3) represents the fact that at least one algorithm should obtain the best result for a region. This indicates that such region has a mixture of features values for which at least one algorithm is the most appropriate. The second line in Eq. (3) indicates that the sum of the F-values for each processed region m with the best algorithm $m_{best}$ from one image should be equal or larger that the F-value resulting from processing the image as whole by a single algorithm.

In this paper we use the salience map of an image to split the image into regions. The salience uses features relevant to human vision [13] and thus splitting images using salience can be used to separate image into regions that have high and low relevance for human segmentation: image regions determined by salience have high probability to contain symbolically meaningful regions and analyzing them separately means, that each subregion can contain such edges that are significant to a human observer. Moreover, because it is not know what features are the most relevant for image splitting and because salience is computed using a mixture of low level features, salience is therefore a good starting point for image splitting. Consequently splitting an image using saliency can potentially generate regions that contains homogeneous features and thus maximize algorithm performance.

In order to determine subregions’ location and size we first calculate the salience map (Fig. 1(a)). From the salience map we calculate for every row and every column the mean of the intensity and of the entropy. This results in two curves for rows and for columns. The average values of each column and each row are shown in the bottom and right graphs respectively in Fig. 1(b). The entropy of the row and column salience are similarly shown in the top and the left

![Image](a) ![Image](b)

Fig. 1. Example of the image splitting algorithm using the salience and the entropy of the salience.
graphs respectively of Fig. 1(b). The regions are created by separating the image by lines drawn at the points of the mean of the salience intensity of the columns and rows curve. One can write, that a line will be drawn at a location \( r \) of the mean salience if the value of the mean at location \( r \) is between lower and higher values of a dynamically calculated bound. In other words, \( p(r) = 1 \) means that a line is drawn at location \( r \) and is given by:

\[
p(r) = \begin{cases} 
1 & \text{if } \min \delta + \delta_{\min} < \text{val}(r) < \max \delta - \delta_{\max} \\
\text{skip} & \text{o.w.}
\end{cases}
\]  

(4)

with \( \delta_{\min} \) and \( \delta_{\max} \) are values obtained in the following manner:

\[
\delta_{\min} = \min(\text{avg}(s(r_i))), i \in [r_1, r_{\text{max}}] \\
\delta_{\max} = \max(\text{avg}(s(r_i))), i \in [r_1, r_{\text{max}}]
\]  

(5)

where \( r_i \) is \( i^{th} \) row or column, and the \([r_1, r_{\text{max}}]\) is the range of the pixels in \( r \).

The meaning of the splitting at the given locations is that we desire to always split a given image to regions of low or high interest. The lowest and the highest salience values should not be used as places to split the image because the highest salience value regions are of interests and the lowest salience areas can be used to determine boundaries of highest area of salience. Thus the image is split by searching for the sharpest gradients in the changes in the row and column salience.

As can be seen by transposing the lines from Fig. 1(b) onto Fig. 1(a), the salience based splitting results indeed in regions that separate the image into areas containing homogeneous regions of salience intensity.

A concern that arises from a regional image processing is the decrease of the quality of the result as a function of the region size. This is a natural consequence of the size factor; various multi-level algorithms use different higher level strategies to remove noise, to reduce the intensity level of boundaries or to determine where to put the edge. Table 2 shows the evaluation of a set of images using regions of various size. Observe that the increasing size of the regions, the result quality of the boundary detection increases as well.

<table>
<thead>
<tr>
<th>Region min-limit</th>
<th>F measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.59</td>
</tr>
<tr>
<td>10</td>
<td>0.61</td>
</tr>
<tr>
<td>20</td>
<td>0.62</td>
</tr>
<tr>
<td>30</td>
<td>0.64</td>
</tr>
</tbody>
</table>

The reason that smaller regions generate images with lower quality of the boundary is mainly due to the increased noise concentration. Moreover, having many regions per image, the resulting automated selection of the algorithms error’s increases with the increasing number of image regions. Thus, to avoid splitting the image into regions of very small size, the splitting procedure is parameterized by a proximity coefficient \( \gamma \) that determines the minimal size of the region. For instance \( \gamma = 30 \) means that a region cannot be smaller than \( 30 \times 30 \) pixels.

A natural question that occurs is that despite the salience being a combination of several low level features when and in what case other possible image features are better suited for image splitting.

Another shortcoming of the proposed image splitting method, is that the resulting regions do not allow to extract precise regions that contains only the region of interest. On one hand this is a negative effect because the extracted sub-images contains such regions that are not interesting or related to the task at hand. This can result in sub-images containing multiple salient regions that would require separate processing. Such regions decrease the possibility of the high quality results as well as the possibility to select the best algorithms because the salient sub-regions might require different algorithms to create best results. On the other hand selecting regions that includes intersections of multiple high saliency regions, allows to obtains local boundaries with a higher intensity.

The bottom line is that the image splitting method is far from being perfect but improves the quality of algorithm selection using machine learning. For instance, the learning result for algorithm selection using the image regions is approximately 70–75% correct. The same approach based on the selection of algorithms using whole images is at maximum 65%. This discrepancy is not only because of the difference of the amount of information between an image and a subregion but also because the amount of the training and testing data for whole images is much smaller and thus does not allow the same quality of learning.

5. Learning the Algorithm Selection

5.1 Cost of algorithm selection

The main desired characteristic of the features used to select algorithms is the distinguishability of the different algorithms. On one hand, using features completely different from features used in any available algorithms allows to objectively select among algorithms. This criterion is based on the fact that in order to maximize the ratio result score/
processing time the features used to distinguish the algorithms should be as independent as possible. In such a case when features used for algorithm selection are different from features used by any of the algorithms the evaluation can be compared on a constant basis. On the other hand, using features from the algorithms to select which algorithm is the most appropriate would save processing time and resources. The ideal situation would be such that each algorithm would operate on the same set of features by applying a different high level strategy. But for a general approach and as will be seen a mixture of features used and additional features provides the best distinguishability.

Table 3 shows the difference between the required processing for algorithms that contain distinguishing features and that does not. The first column shows if the FL set of features used for algorithm selection is contained within the set of FH features used by the algorithms selected for processing. The second column shows the feature requirements of the algorithms available for selection and the last column shows the feature requirements for the processing and the selection. The most desired type of selection is such that minimizes the amount of features: this is shown in the first row of Table 3.

<table>
<thead>
<tr>
<th>Features relation</th>
<th>Processing requirements</th>
<th>Processing and selection requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>FL ( \subseteq ) FH</td>
<td>( F_H - F_L )</td>
<td>( F_H )</td>
</tr>
<tr>
<td>FL ( \nsubseteq ) FH</td>
<td>( F_H )</td>
<td>( F_L + F_H )</td>
</tr>
</tbody>
</table>

5.2 Features for algorithm selection

In order to successfully select an algorithm a set of distinctive features must be identified. The features we used are shown in Table 4; the left columns shows the feature name while the right column shows the size of each feature in number of double precisions coefficients.

Table 4. The features used in the described experiments for algorithm selection.

<table>
<thead>
<tr>
<th>ID</th>
<th>Features</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>sum of gray histogram</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>sum of fft</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>gabor filter</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>wavelets</td>
<td>400</td>
</tr>
<tr>
<td>5</td>
<td>contrast</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>acutance</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>gist</td>
<td>940</td>
</tr>
<tr>
<td>8</td>
<td>orient gradient</td>
<td>400</td>
</tr>
<tr>
<td>9</td>
<td>line feature</td>
<td>120</td>
</tr>
</tbody>
</table>

This initial set of features is as large as possible and is meant to be exhaustive. Thus low level local (oriented gradients, gabor, wavelets) and global features (FFT, intensity histogram), middle level (line features) and also higher level features (gist) are included. The line features is a particular type of middle level feature that combines the line length, line salience and line intensity variation. Computing the average length of lines, average salience of lines and the ratio of the average salience over the average line length allows to estimate the degree of difficulty of extracting a line of certain length at a given intensity threshold. This feature is thus directly related to the boundary extraction problem. All the features used have been normalized (obtained from a normalized image) and stretched to such range that the feature values of the test set are contained in the range of the train set feature values.

To determine which features are well suited for algorithm we compared the predictive quality available features. For this we evaluated the individual features and combinations of them for the learning of algorithm selection. The learning has been performed using the Ada-boost and the SVM packages implemented in Matlab. The input images are taken from the BSDS database 500 [3], where the training set consists of 200 images and the testing set of 200 images as well.

Table 5 shows comparative evaluation of three features Intensity Histogram (Hist), Gist and the line feature (Line).

The first column shows the name of the feature and the second column shows the feature’s power predictability in percent when used on whole images. The third column shows each feature’s predicting power when used an image regions. Notice that while for the whole images the sum of all features provides the best predictive quality (45.45%), in the case of image regions it is not the case (62.12%). This is interesting because it confirms the fact that splitting the image using the salience indeed separates natural images into simpler regions that can be learned with less features and higher accuracy.

Observe that the Table 5 shows that certain features are better than others, that some combinations are better than others and finally that using all features does not necessarily allows for the best performance.
To select the best features various methods exist. Here we tested various Filters feature selection methods: Filters are such selections that select the features as a preprocessing step independently of the algorithm used for prediction. For instance well known methods such as Fischer Score [10], Correlation [28], mRMR [20], Mutual Entropy [7], Matrix Factorization (LDA, SVD) and so on have been used in various feature selecting contexts. We experimentally determined that for the task at hand the best method turned out to be the Fisher’s Score as well as analysis using the LDA search.

6. Image Reconstruction

To obtain the resulting contours from the resulting regions two issues are addressed as the result of processing the image by regions.

1. Depending on the size and on the type of region being processed some results of the regional processing must be enhanced by increasing the overall intensity of the edges. This is critical in order to obtain a high quality of output because the edges must be homogeneous as much as possible. The reason for this requirement, is the threshold evaluation; if the desired boundaries do not have homogeneous intensity, the threshold of intensity to obtain the whole boundary can be too low and thus a lot of noise would be present in the image as well. This results in a low score of the image as the idea behind the thresholded evaluation is to find such threshold that would maximize the pixels belonging to the desired boundary and minimize the amount of pixels representing the noise. The algorithm for equalizing the image regions intensity first determines the maximum and minimal difference between each of the adjacent regions. An example of such processing is shown in Eq. (6). Each row in Eq. (6) corresponds to one region, and each column corresponds to down, up, left and right neighbors. Each coefficient shows the maximal (or minimal) difference between neighboring pixels in adjacent regions.

\[
\begin{bmatrix}
0 & 0.0706 & 0 & 0 \\
0.1333 & 0.0706 & 0 & 0 \\
0 & -0.1333 & 0 & 0 \\
\end{bmatrix}
\]  

(6)

A negative coefficient indicates that the current region needs to have their intensity increased while a positive coefficient indicates that the region’s intensity needs to be decreased. Using this information the algorithm then calculates for each of the region the amount by which its intensity should be increased or decreased using the following formula:

\[
\text{coeff} = \begin{cases} 
\frac{1}{1 + \text{coefficients}(i)} & \text{if coefficient}(i) < 0 \\
\text{coefficients}(i) & \text{o.w.} 
\end{cases}
\]  

(7)

Each region is multiplied by the appropriate coefficient and the resulting image is iterated over using the same approach until the whole image is homogeneous. This is given by either stopping the iteration when the differences between regions becomes small enough (< \(\delta\)) or when the iterations do not alter the contour condition matrix shown in Eq. (6). The final image is then constructed by putting the regions side by side according to their coordinates.

2. As shown in Section 4 the processing of regions results in increased noise size. To remove some of this additional noise, an image analysis based on regional correspondence and a cellular automata are used.

(a) The regional analysis uses the concept that the amount of lines in a human segmentation is in general very low. In fact, looking at the BSDS300 data training set for instance we can determine that the amount of pixels in the boundary used to segment of the image is between 0.5 to 4% of all the pixels in the image (Fig. 2). This means that when generating a boundary a certain amount of pixels can be removed as the amount of pixels generated by different algorithms is at least 40%. To remove some of the noise pixels we first determine areas of the boundary image obtained by processing regions by the selected algorithms. This is achieved by determining regions with high amount of border pixels. In such areas we determine also the number of terminal pixels: pixels that are at the end of boundary lines. In areas with noise such pixels are in relatively

<table>
<thead>
<tr>
<th>Feature</th>
<th>Full images</th>
<th>Regions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hist</td>
<td>18.18</td>
<td>58.18</td>
</tr>
<tr>
<td>Gist</td>
<td>24.24</td>
<td>57.58</td>
</tr>
<tr>
<td>Line</td>
<td>39.39</td>
<td>63.94</td>
</tr>
<tr>
<td>Hist+Gist</td>
<td>33.33</td>
<td>56.97</td>
</tr>
<tr>
<td>Hist+Line</td>
<td>36.36</td>
<td>56.67</td>
</tr>
<tr>
<td>Gist+Line</td>
<td>42.42</td>
<td>65.45</td>
</tr>
<tr>
<td>All</td>
<td>45.45</td>
<td>62.12</td>
</tr>
</tbody>
</table>
high concentration and thus such areas can be determined by comparing the density of terminal pixels in different regions of the image. Regions with high amount of such terminal points in a small area can be removed.

(b) A cellular automaton is used to increase the consistency of longer boundaries and remove shorter boundaries. For this we first start from a thresholded boundary image. The threshold is set adaptively on a per image basis so that the pixels with the lowest intensity are filtered out. Then a cellular automaton with a mask showing in Eq. (8).

\[
M = \begin{pmatrix}
1 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 1 \\
\end{pmatrix}.
\]  

Equation (8)

Each pixels that obtain a coefficient of 3 or larger has its intensity diminished while pixels with smaller coefficient than 3 has its intensity increased. The general result is that most of the single lines have intensity increased however because of the lines are rarely uniform in both intensity and in the amount of neighboring pixels linger lines are often interrupted. To solve this problem, a local line growing algorithm using gradient to complete interrupted lines.

7. Experiments and Results

In order to evaluate the algorithms we performed an exhaustive processing of all the subregions of all images from the testing and training data sets by available algorithms. The algorithms that we used are shown in Table 6 and the machine learning of the algorithm selection mechanism was done for comparison using two algorithms: SVM and Ada-boost.

<table>
<thead>
<tr>
<th>Algorithm name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph cut [4, 5]</td>
</tr>
<tr>
<td>Normalized cut [24]</td>
</tr>
<tr>
<td>ROI [8]</td>
</tr>
<tr>
<td>Salience [9]</td>
</tr>
<tr>
<td>CGTG [16]</td>
</tr>
<tr>
<td>BGTG [16]</td>
</tr>
<tr>
<td>Global probability of boundary [15]</td>
</tr>
<tr>
<td>Improved global probability boundary [3]</td>
</tr>
</tbody>
</table>

Table 6. The algorithms used in the experiments.

To train the Ada-boost and the SVM algorithms, we first generated the ideal results by splitting each image into sub regions and then selecting the algorithm with the highest the F value for each region by exhaustive search. This resulted in both the training and the testing data sets. Each region was then transformed into a set of features from Table 4 and the training set was used to train the Ada-boost/SVM algorithm. The verification was performed over the regions from...
the test set. To evaluate the algorithms we compared the result of the contour generation and splitting algorithm to the performance of every algorithm alone generating contour for the test data set.

Figure 3 shows the result of the learning using the Ada-boost. Observe that the learning result is extremely poor and does not allow any induction. Interestingly, the distribution of the generated labels using the Ada-boost is quite similar to the desired labels. The distribution of labels from the training and testing data set and from the estimated labels for the training and the testing set (Fig. 4).

In order to improve the result observe that certain labels (algorithms) present in the training and testing data set are very under-represented. This is shown in Fig. 4. There are five algorithms that are present in quantity that is learnable while three algorithms are mostly present as noise (less than 5% of samples). To learn more precisely a larger data set is required so as to obtain more samples of such regions that corresponds to these minority algorithms.

To improve the learning with the available data, we remove these algorithms from the training and the testing set. Also, we use SVM and Ada-boost to compare which one performs better. Using this reduced data sets the resulting learning accuracy became 40% using Ada-boost and 70% using the SVM. Observe that even with removing the minority regions, the learning remains not very good. However this learning can be used to evaluate the quality of region based processing. The results of this learning are visualized on Figs. 5(a)–5(c). Figures 5(a) and 5(b) shows the shows the learned label distribution from both the training and testing data sets. Figure 5(c) shows the result of learning with respect to the number of weak learners. Notice that the removal of the minority algorithms improved the learning by more than 60%.

Table 7 shows the comparison between some of the algorithms F values run on the BSDS500 benchmarks. Note that the values might differ from same benchmarks reported by other papers as we run the algorithms as they are without any optimizations or improvements. Observe that on average our method is not better than the best algorithms however looking closer on the benchmarks the algorithm selection before normalization is better in 10% of the cases while after the normalization and the noise removal it is better in 17% of the cases. This results is due to several factors:

1. The image splitting to regions and the consequent algorithm assignment is not perfect because the problem if selecting the most appropriate regions is still an open issue.
2. The learning of algorithms selection requires better learning rate.
The noise removal is not optimal and often removes areas of the image that contain not only noise but also the desired boundaries. Figure 6 shows the comparison between selected images contours generated by the best single algorithm, the ideal selection obtained by exhaustive search and the obtained contours using the learned model of the Ada-boost machine learning. Note that the selected benchmarks show some of the cases where the learning was able to improve or find the best available algorithm selection.

Several facts can be observed from the obtained results.

1. The ideal region-based processing by algorithm selection improves the overall contour detection in cases where the images have very different regions in terms of the features that are present. This means that indeed the processing of an image using the best algorithm per region is a viable approach for robust processing (Fig. 6).

2. The result of the learning is quite poor with 40 to 70% of error rate but using the resulting model on the test set still obtains better result than any of the single algorithms. This is because, the training data set is highly not regular. This can be improved by generating a larger dataset or by scaling the available data set by iterative sampling.

3. As can be seen in Fig. 6 even with the relatively poor results of the machine learning certain cases can be improved by a considerable value of the F measure.

8. Conclusions and Future Works

In this paper we presented a study into the selection of segmentation algorithms for natural images. The present work
is focused on algorithm selection but further investigation into the feature set selection and feature set reduction is planned. We showed that in some cases our approach allows to improve the contour detection quality when regions’ features can be selected in such manner that maximizes performance of some algorithms. The future work includes

![Fig. 6. Comparative display of results, the first column shows the original input image, the second column shows the contours generated by the algorithm from [3], third column shows the contours obtained by algorithm selection using the trained SVM model and the fourth column shows the ideal contours obtained by the exhaustive search. Each result shows the calculated F measure for the obtained contours.](image)
a more robust approach allowing to select when region splitting and algorithm selection can be useful. Moreover, particular the features must be properly minimized to a set that would provide the best selection for a minimal computational effort and the image splitting and merging must be improved to be comparable to the best available segmenters. Thus combining features and proper features selection using algorithmic approach are to be explored. Finally the noise removal must be improved to precisely target the areas of noise and avoid removing boundary lines. This can be achieved by a better analysis of the image and a more adaptive noise removal.

REFERENCES