Image Segmentation via Fusion Neural Networks

Sih-Yin Shen, Ya-Yun Jheng, Chun-Shun Tseng, and Jung-Hua Wang
Electrical Engineering Department, National Taiwan Ocean University
2 Peining Rd. Keelung, Taiwan
e-mail: jhwang@mail.ntou.edu.tw

Abstract—This paper presents a self-organizing fusion neural network (SOFNN) which is effective in performing fast image segmentation. Based on a counteracting learning strategy, SOFNN employs two parameters that together control the learning rate in a counteracting manner to achieve free of over-segmentation and under-segmentation. Regions comprising an object are identified and merged in a self-organizing way, and the training process will be terminated without manual intervention. Because most training parameters are data-dependent, implementation of SOFNN is simple. Unlike existing methods that sequentially merge regions, all regions in SOFNN can be processed in parallel fashion, thus providing great potentiality for a fully parallel hardware implementation. In addition, not only the immediate neighbors are used to calculate merging criterion, but the neighboring regions surrounding the immediate regions are also referred. Such extension in adjacency helps achieve more accurate segmentation results.

Keyword: Neural networks, image segmentation, extended adjacency, counteracting learning, watershed Analysis.

I. INTRODUCTION

Image segmentation plays a preliminary yet essential role in most image analysis tasks. Improper or inaccurate segmentation often make impossible the follow-up recognition or analysis tasks. The goal of segmentation is to identify objects of interest that satisfy certain pre-defined homogeneity criteria, whereas the primary objective of clustering [1] is to partition a given set of data or objects into clusters (subsets, groups, classes). This partition should have the following properties: 1) Homogeneity within the clusters, i.e., data that belongs to the same cluster should be as similar as possible and 2) Heterogeneity between clusters, i.e., data that belongs to different clusters should be as different as possible. This concept has been found useful [2, 3, 4] in realizing image segmentation wherein a raw image is taken as the input data, then clustering operates on the feature space to label sufficiently similar and adjacent pixels. After the clustering process, the labeled results are mapped onto the image plane to obtain the final segmented objects.

On the other hand, in hybrid-based segmentation methods [5], a pre-process is initially invoked to dissect the input image into a set of small primitive regions referred to as initial image tessellation. One well known such pre-process is the morphological watershed transform [6, 7] which has drawn great attentions from researches for two main reasons: its easiness in implementation, and more importantly, it generates a completely closed contour. However, due to its nature of dividing input image according to its belief, there exists the so-called over-segmentation problem, namely, too many small primitive regions.

There has been a great deal of literature aiming to solve the over-segmentation problem, e.g. the fast nearest neighbor region merging graph (NNG) [8]. But they adopt sequential strategy [9, 10] in that the two most similar regions are searched globally for merging. This inevitably requires a time-consuming sorting procedure, not to mention that after a region pair is merged the graph structure needs to be updated. Another shortcoming of sequential merging is the lack of spatial information needed for estimating the merging criterion, resulting in false contours and less accurate segmentation. Obviously, this undesired effect will be further deteriorated by the more number of initial primitive regions.

To deal with the aforementioned problems and to further improve merging accuracy, we propose a novel approach called self-organizing fusion neural network (SOFNN) to achieve image segmentation. Unlike k-means algorithm [4], SOFNN does not need to pre-specify the final number of clusters. Initially each primitive region is taken as an individual neuron, and based on similarity and adjacency measures regions are clustered in feature space. Like other neural network models, SOFNN simultaneously updates weights of neurons, which greatly improves the computation efficiency over the sequential merging. Another essential feature of SOFNN is the association of each target neuron $R_i$ with two statistical parameters $\alpha_i$ and $\beta_i$, each favors merging and splitting, respectively. During the training process, local statistics are collected to calculate $\alpha_i$ and $\beta_i$, which in turn are used to adjust the learning rate. With the counteracting effect between $\alpha_i$ and $\beta_i$, the learning dynamics can be stabilized and both over-clustering (corresponding to over-segmentation) and under-clustering (under-segmentation) can be prevented.

Furthermore, in order to further improve segmentation accuracy, we propose the extended adjacency strategy and use it to update the similarity measure for each $R_i$. The strategy emphasizes that not only the immediate regions of the target region but their neighboring regions are also accounted for. After updating, features of regions will be blended to each other, hence the name fusion. Regions satisfying a well-defined statistical criterion will be merged.
II. Self-Organizing Fusion Neural Networks

A. Fusion in the feature space

To identify regions that are qualified to be merged in the next iteration, it is necessary to keep track of adjacency information changed due to merges in the preceding iteration. An ideal choice of data structure would be the undirected Region Adjacency Graph (RAG) [10]. In fact, all the statistical parameters used in this work are closely related to the adjacency information. The interpretation of an edge in RAG is this: the more two adjacent regions similar, the more value the edge is. After RAG is built, each target region \( R_i \) is characterized by the feature \( w(R_i) \) defined as the average intensity of all pixels in \( R_i \), value of which will be repeatedly updated. In SOFNN, we use \( w(R_i) \) as the weight associated with a neuron. The similarity measure at iteration \( t \) between an arbitrary region \( R_i \) and its immediate neighboring regions \( R_j, \{R_j, j=1, 2, ..., n-1\} \) is prescribed as

\[
E_{ij}^t = \frac{1}{e^{\lambda(t) \cdot (w(R_i) - w(R_j))}} , 0 < E_{ij}^t \leq 1. \tag{1}
\]

where \( n \) is the number of regions in \( N(R_i) \), and \( \lambda(t) \) is a time-varying function defined as

\[
\lambda(t) = \lambda_0 e^{{\tau}(t-1)} \tag{2}
\]

The parameter \( \tau \) in Eq.(2) is a time constant. If otherwise specified, \( \lambda_0=0.3 \) and \( \tau=0.2 \) are used throughout. Note that because \( w(R_i) \) of all regions are time-varying, the similarity defined in Eq.(2) is also time-varying.

Although Eq.(1) quantitatively measures the similarity between two adjacent regions, using it alone is not adequate in practical implementations. The main difficulty lies in the fact that regions subjectively perceived by human eyes as belonging to an object might not possess the similarity sufficient to be identified as the same object through a thresholding scheme in ordinary computer algorithms.

In fact, a universal threshold for sufficient similarity is very difficult, if not impossible, to be specified. Thus, instead of using a pre-fixed [11] threshold, we employ a dynamic merging criterion based on a homogeneous measure using local statistics. Thus, by allowing regions’ features to be simultaneously updated each iteration and checked if some of them have become homogeneous, the task of image segmentation can be conducted without the difficulty of setting a global threshold. Another aspect of this approach is that regions comprising an object are identified in a self-organizing way, and it will terminate without manual intervention. This is an especially desirable feature in applications where specifying the number of objects in the final result is either difficult or unnecessary.

The simultaneous updating in the feature space actually facilitates the fusion phenomenon useful for identifying regions comprising an object. The update equation for \( w(R_i) \) is given as

\[
w^{t+1}(R_j) = w^t(R_j) + \epsilon_i^t \left[ w^t(R_i) - w^t_{av}(N(R_j)) \right] \tag{3}
\]

where \( w^t_{av}(N(R_j)) = \frac{\sum_{R_j \in N(R_j)} w^t(R_j)}{n} \). \( N(R_j) \) denotes the set of adjacent regions surrounding to \( R_j \) in RAG, including \( R_j \) itself. \( N(R_j) \) is also called the immediate neighbors of \( R_j \). The value of \( w^t(\cdot) \) is iteratively updated. We will elaborate the learning rate \( \epsilon_i \) in the next section.

The beauty of iteratively applying Eq.(3) is that it can facilitate systematic identification of which adjacent regions to be merged. After the watershed operation, the primitive small regions that actually belong to an object should be quite similar in a local sense. Generally, use of Eq.(3) tends to force the region and its adjacent regions to become more alike. That is, the weights of adjacent regions become closer in feature space, collectively appearing as a fusion phenomenon taking place among the adjacent regions. With the fusion effect, very soon the features of adjacent regions become similar enough to be identified as candidates for merging. After some beginning iterations, updating should be made more gradual by increasing \( \lambda(t) \) in order to avoid regions with distinct dissimilarity are erroneously merged into a bigger region.

Following the update, adjacent regions are checked if they are qualified as the comprising components of an object. If so, they will be merged into a bigger region. Just like the weights of all regions are simultaneously updated, the region merging in SOFNN can be conducted in parallel fashion. In subsection C we will discuss the merging criterion in details.

B. The Counteracting Learning

To proceed, we first define a statistical parameter \( \alpha_i \) as the number of adjacent regions that “see” \( R_i \) as their most similar region. The initial value of \( \alpha_i=1 \) for \( i=1,\ldots,n' \) (\( n' \) is the number of regions at iteration \( t \)). The rationale of using \( \alpha_i \) is best explained with the directed similarity flow diagram in Fig.1. The directed arrow, say, the one pointing from \( R_1 \) to \( R_1 \) represents that among the adjacent regions of \( R_1 \), namely \( \{R_1, R_2, R_3\} \), \( R_1 \) is seen by \( R_1 \) as the most similar region, i.e., \( E_{1max}^{\alpha_{i}=\alpha_{j}=1} = Max \{E_{11}, E_{12}, E_{13}\} = E_{11} \). On the other hand, besides \( R_1 \) itself there are three regions \( \{R_5, R_6, R_6\} \) that see \( R_1 \) as their most similar region, thus \( \alpha_i=1+3=4 \). The learning rate \( \epsilon_i \) of region \( R_i \) is prescribed as

\[
\epsilon_i^{t+1} = \epsilon_i^{t} \times \exp \left[ -\left( \frac{\beta_i^{t} \times \alpha_i^{t}}{\alpha_i^{t}} \right) \right] \tag{4}
\]

\[
\alpha_i^{t+1} = \sum_{i=1}^{n'} \alpha_i^{t} \quad \text{and} \quad \beta_i^{t} = \sum_{i=1}^{n'} \beta_i^{t}
\]

where \( \epsilon_i^{t} \) is the initial learning rate. If otherwise specified, \( \epsilon_i^{t}=0.04 \). Examining Eq.(4) clearly indicates that a larger \( \alpha_i \) (21) will increase the learning rate. That in turn, according to Eq.(3), encourages the fusion and merging to occur.
Another parameter $\beta_i$ in Eq.(4) provides, however, just the opposite effect. The rationale of $\beta_i$ is explained as follows. Consider the case when $R_3$ is distinctively different from its surrounding regions $\{R_1, R_2, R_4\}$, namely $E_{13}, E_{23}, E_{43}$ are rather small, in this case it is desired to inhibit the merge of $R_3$ to $R_1$. Nonetheless, as described previously, $\alpha_i$ will be increased by 1 since $\text{Max}(E_{13}, E_{23}, E_{43})=E_{13}$, no matter how small values of $E_{13}, E_{23}, E_{43}$. Therefore, in order to counteract the effect caused by $\alpha_i$ and prevent under-segmentation, another statistical parameter $\beta_i$ defined as the sum squared differences between $R_i$ and its adjacent regions, i.e. $\sum [w(R_i) - w(R_j)]^2$ is incorporated into Eq.(4). The larger $\beta_i$ is, the smaller the learning rate $\varepsilon_i$ will be. The use of $\beta_i$ advantageously prevents regions with distinct dissimilarity from being merged, hence avoiding under-segmentation.

Using the input image “peppers”, Fig.2 shows values of $\alpha_t / \alpha_{av}$ and $\beta_t / \beta_{av}$ which are defined as the average of $\alpha_t$ and $\beta_t$ over all regions in iteration $t$. We can see that $\beta_{av} / \beta_T$ is monotonically increased, in contrast, $\alpha_t / \alpha_{av}$ always decreases. The result clearly demonstrates that $\alpha_i$ and $\beta_i$ together in Eq.(4) has the effect of avoiding over-segmentation and under-segmentation.

C. The Merging Criterion

To conduct merging, an important parameter is introduced as follow

$$\zeta^{t+1}(R_i) = \frac{1}{n} \sum_{R_j \in N(R_i)} w^{t+1}(R_j)$$

For an arbitrary region $R_i$, besides the intra-regional parameter $w(R_i)$ that measures the average intensity of pixels in $R_i$, the inter-regional parameter $\zeta(R_i)$ is useful in characterizing overall intensity of area surrounding $R_i$. Thus, the value of $|\zeta(R_i) - \zeta(R_j)|$ in a sense stands for the dissimilarity (in feature space) between two areas centered at $R_i$ and $R_j$, respectively. And the mean difference between and those of regions adjacent to $R_i$ is written as
\[ d^{t+1}(R_i) = \frac{1}{n-1} \sum_{R_j \in N(R_i)} |\epsilon^{t+1}(R_i) - \epsilon^{t+1}(R_j)| \]  

(6)

Note that, with \( \zeta(R_i) \) prescribed in Eq.(5), the calculation of \( d(R_i) \) actually takes into account the adjacent regions of \( N(R_i) \) too, namely, neighbors’ neighbors, not just immediate neighbors of \( R_i \). We denote the regions participating in evaluating \( d(R_i) \) as \( S(R_i) \), that is, \( S(R_i) \) is the set of neighbors of immediate regions \( N(R_i) \) of target region \( R_i \), \( N(R_i) \subset S(R_i) \). In this work, \( d(R_i) \) used as an effective statistical gauge to judge whether a region adjacent to \( R_i \) comparing to the rest of adjacent regions is more qualified to be merged. As will be shown later, this extension in the effective adjacent area of \( R_i \) helps reduce the likelihood of erroneously merging two regions, and makes great difference to the merging result. Statistically speaking, any adjacent region of \( R_i \) with \( w(R_i) \) similar to that of \( R_i \) to such extent that their difference is less than \( d(R_i) \), it represents that \( R_i \) is more similar to \( R_i \), compared to most of other adjacent regions. Thus, the rule of identifying an adjacent region \( R_i \) is simply this:

\[ w^{t+1}(R_i) - w^{t+1}(R_j) \leq d^{t+1}(R_i), \text{merge } R_i \text{ and } R_j \]

(7)

The rationale of using \( d(R_i) \) in Eq.(7) as the local threshold is explained as follows. Because \( S(R_i) \) covers not only immediate adjacent regions of \( R_i \) (i.e., \( N(R_i) \)), but also the adjacent regions of \( N(R_i) \), the value of \( d(R_i) \) in a way stands for the variation of intensity across the area \( S(R_i) \) centered at \( R_i \). For an area \( S(R_i) \) comprising lots of smooth regions, the corresponding values of \( d(R_i) \) should be relatively small, regions in that area are more easily to be merged. In contrast, an area \( S(R_i) \) has large values of \( d(R_i) \) would indicate intensity variations in \( S(R_i) \) are significant, implying the existence of separated objects in \( S(R_i) \) is very likely, merging regions in \( S(R_i) \) is certainly not encouraged.

After a larger region is generated due to merging, its weight will be replaced with the average weight of the component regions. And regions that were not merged at current iteration will use the updated weights (via Eq.(3)) in the subsequent iteration. Thus, except in the first iteration calculating \( w^{t+1}(R_i) \) is region-based, rather than pixel by pixel. Note that because the parameter \( d(R_i) \) is calculated on region base, not pixel by pixel. Thus, calculation of \( d(R_i) \) only uses a small portion of the total computation load. Furthermore, if some regions and their adjacent regions are not merged at present iteration, their average discrepancy parameters will not be recalculated in the subsequent iterations. This also helps lessen a couple of computation load, especially in the case of huge number of initial regions. In fact, the speed of fusion is fast during the beginning iterations, but as more regions are merged, it will slow down as the number of identified objects become close to the number of actual objects in the image. This property is useful as it provides a plausible way of setting the terminating condition for the SOFNN algorithm. Namely, the iterative operation will be terminated when there are no more regions to be merged.

It is not unusual to see that some regions get merged and some don’t at certain iteration. To proceed to the next iteration, the updated values of weights, similarity, and RAG will be recalculated for the merged regions along with the intact regions. In short, SOFNN mainly consists of three phases: counteracting learning, fusion, and merging, they perform image segmentation in a manner of relay race.

III. EXPERIMENTAL RESULT

We use the benchmark image “peppers” (256×256, 256 gray levels) shown in Fig.4 (a) as the input. The result after applying watershed pre-process is shown in Fig.4 (b). The over-segmentation phenomenon is quite vivid as the total number of initial regions is 1566. After applying Eq.(3) to blend the working features of adjacent regions. Larger regions are generated by the merging process using Eqs.(5), (6) and (7). And Fig.4 (c) illustrates the 179 objects painted with weights updated at the 1st iteration. After termination at iteration 4, Fig.4 (d) shows the final result of 103 merged objects painted with their corresponding original pixel intensities, the total computation time is 0.283 sec. It is reasonable to conjecture that the parallel fashion indeed drastically reduce the computation time. Fig.5 shows the merging rate in “peppers”. Correspondingly Fig.6 (a) shows the value of \( \varepsilon \) (defined as the average of \( \varepsilon_i \) over all regions at the current iteration) in successive iterations. Obviously, we can see that the learning rate decreases as a whole to achieve stability but can slightly oscillate at times to avoid trapping in a local minimum.

![Fig4](image_url)
IV. CHARACTERIZATION

Results in Section III have shown that the proposed SOFNN indeed performs well. We conjecture its effectiveness roots in the extended adjacency which offers sufficient neighborhood information to the target region in measuring similarity degree. Each threshold is evaluated by taking into account the weight of immediate neighbor as well as their adjacent regions. It provides a more objective spatial measure for target region in judging whether to merge its adjacent regions.

In conventional segmentation algorithms where the basic unit to be processed is one region or a pixel, most often the immediate neighbors are used for measuring similarity/homogeneity. In contrast, SOFNN explores the inter-regional information; and the merging process is actually conducted in unit of regional cluster, not region nor pixel. That is, the local threshold is obtained by examining not only the immediate neighbors of target region but the neighboring regions surrounding them. This attribute not only offers much more adjacency information to the target region, but provides more tolerance to input noise. Two types of noise are tested, namely, impulsive (salt-and-pepper) and Gaussian. Fig. 7 (a) is the test input “house”. Fig. 7 (b) and (c) is corrupted by 20% salt-and-pepper noise, and Gaussian noise (zero mean and variance=0.01), respectively. Fig. 7 (d) is the tessellation of Fig. 7 (c) (after filtering by a mean filter and watershed analysis).

Using the input image corrupted by the salt-and-pepper noise (after applying a 3×3 median filter), Fig. 7 (e) and (f) show the segmentation result via SOFNN and SRM [12], respectively. In the case of Gaussian noise, Fig. 7 (g) and (h) shows the results via SOFNN and SRM, respectively. We can easily spot several false contours in Fig. 7 (f) and (h). This result verifies that SOFNN indeed is robust to input noise.

It is worthy noting that although proper extended adjacency information can provide an objective local threshold, excessive extension could incur a meaningless threshold such as a nearly globe threshold that certainly cannot work as a “local” threshold. This is analogous to an excessive mask size in a mean filter.

Fig. 6 (a) the average learning rate $\varepsilon_{av}$ of each iteration (b) the zoom-in of Fig. 7 (a) during the beginning 20 iterations

(a)                        (b)
(c)                       (d)
Fig. 7 (a) input image “house” (b) corrupted by 20% salt-and-pepper noise (c) corrupted by Gaussian noise with mean=0 and variance=0.01 (d) output of mean-filtering and watershed analysis; (e) segmented result via SOFNN using (b) after applying a 3×3 median filter as input (f) segmented result via SRM (g) segmented result via SOFNN using (c) as input (h) segmented result via SRM.

V. CONCLUSIONS

We have presented a new approach SOFNN that incorporates simultaneously feature fusion, counteracting learning, and extended adjacency strategy to solve the problems of over-segmentation that plagues conventional approaches. To improve the segmentation accuracy, the proposed approach employs a pair of counteracting parameters to avoid under-segmentation. In addition, results have shown that the strategy of extended adjacency effectively acquire superior local information between the target region and its adjacent regions for determining the merging criterion. The strategy is effective and robust even when noise is in presence. Note that regions comprising an object are identified and merged in a self-organizing way, and the training process will be terminated without manual intervention, namely users need not pre-specify the terminating number of objects. Another major advantage of SOFNN is that all regions are processed in parallel fashion, thus providing great potentiality for a fully parallel hardware implementation.

ACKNOWLEDGMENT

This research was supported by the National Science Council of Taiwan under grant NSC 94-2213-E-019-014.