Fast Image Segmentation using Region Merging with a k-Nearest Neighbor Graph

Hongzhi Liu*†, Qiyong Guo*, Mantao Xu†, I-Fan Shen*
*Department of Computer Science and Engineering
Fudan University, Shanghai, China
Email: flamephenix@gmail.com, {qyguo, yfshen}@fudan.edu.cn
†Global R&D Center, Carestream Company, No.27, New Jinqiao Road, Shanghai, China.
Email: mantao.xu@carestreamhealth.com

***Abstract*—A fast region merging method is proposed for solving the image segmentation problem. Rather than focusing on the global features of the image, our attention is drawn to local relationship between neighbor pixels with the goal that all similar pixels should be segmented in the same region. In this paper, the image segmentation problem is treated as a region merging procedure. To solve the problem, an initial oversegmentation is performed on the image and a *k*-Nearest Neighbor (*k*-NN) Graph whose vertexes denote regions is built. A new region similarity measure function is also proposed and the region similarity is assigned to the edge as its weight, which can make use of pixel intensity, edge feature, texture and so forth in a unit form. In *k*-NN graph, each vertex chooses exactly *k* nearest neighbors to connect. With it, the computation complexity of merging process can be reduced to $O(\tau N \log_2 N)$; here, τ denotes the number of nearest neighbor updates required at each iteration while N denotes the number of the initial regions. Implementation of the proposed algorithm is introduced, and some experiment results are given to prove our method's robustness and efficiency.**

I. Introduction

Image segmentation is one of the basic problems in image processing and computer vision. The segmentation accuracy determines the eventual success or failure of many existing techniques for image description and recognition [1], [2], image visualization [3], [4], and object based image compression [5], [6]. The goal of image segmentation is to subdivide an image into its constituent regions (sets of connected pixels) or objects, such that each region is homogeneous itself whereas different regions are heterogeneous with each other.

The segmentation problem can be approached by finding boundaries between regions according to discontinuities or using threshold based on the distribution of pixel properties [7]. Many techniques also solve the problem in the way of finding the partitions directly [8], namely region-based segmentation. Its drive is to detect regions which satisfy certain predefined homogeneity criteria. Normally, the input image is first tessellated into a set of homogeneous primitive regions. Then an iterative merging process is applied, within witch similar neighboring regions are merged according to certain decision rules [9]. The key of this method is the region homogeneity definition, usually determined by hypothesis testing [10].

The algorithm proposed in this paper could be considered as one kind of techniques based on regions, which consists of two steps: initial oversegmentation and region merging. Nowadays, many algorithms in morphologic have been proposed to get the primitive regions and most of them are based on watershed transform [11], [12]. However, the results are still not satisfactory enough since the number of initial regions is always too more. Therefore, a good region merging algorithm is needed to compromise the drawback. There are three key points in the merging algorithm design: (a) how to measure the homogeneity between regions; (b) how to merge the regions fast; (c) how to terminate the merging process. Our attention is mainly paid to the first two points. The region homogeneity is commonly defined by global features [10]. Nevertheless, if the image is not in color homogeneity, the results will be very poor with this definition. To overcome this drawback, we define the region similarity by the properties of the pixels along region edge. This is a local definition that can make use of kinds of features in a unit form. As a result, different kinds of images can be segmented in the same framework. Though we only implement the algorithm with color and edge features, the results still indicate a remarkable improvement. In merging process, we use a *k*-nearest neighbors (*k*-NN) graph proposed by Pasi in [13]. It can reduce the complexity of region merging to $O(\tau N \log_2 N)$. Finally, a rule to stop the merging process can be adopted for unsupervised segmentation.

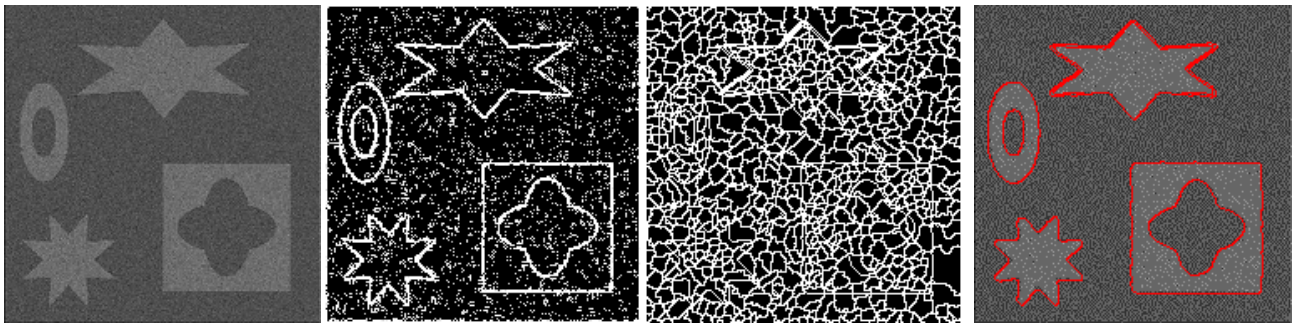
The rest of the paper is organized as follow: Section II gives a segmentation problem formulation, and outlines our proposed algorithm. In section III, we first present our new region similarity definition, and then demonstrate the description of the fast region merging method using a *k*-NN graph. The image oversegment method and how to construct the *k*-NN graph is introduced in section IV. At last the experimental results and conclusions are shown in section V and section VI respectively.

II. Problem Formulation and Algorithm Outline

Let $R = \{p_1, p_2, \ldots, p_N\}$ represents the set of the entire image region, and $p_i (1 < i < N)$ represents the image pixels. We may view the segmentation problem as a process that partitions R into K subregions, $R_1, R_2, \ldots, R_K$, such that

$$\begin{aligned}
&\textbf{(a)}\ R = \bigcup_{k=1}^{K} R_k, \\
&\textbf{(b)}\ R_i \cap R_j = \emptyset, \forall i, j \in \{1, 2, \ldots, K\}, i \neq j, \\
&\textbf{(c)}\ P(R_k) = TRUE, \forall k \in \{1, 2, \ldots, K\}, \\
&\textbf{(d)}\ P(R_i \cup R_j) = FALSE, \forall i, j \in \{1, 2, \ldots, K\}, i \neq j.
\end{aligned} \tag{1}$$

978–1–4244–1674–5/08/$25.00 © 2008 IEEE



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

Fig. 1. Example of segmentation stages. (a) Manual noisy image. (b) Edges detected in (a). (c) Primitive regions with watershed transform (103 regions). (d) Final segmentation ($\sigma_1 = 1$, $\sigma_2 = 0.7$, 7 regions).

Here, $P(R_i)$ is a logical predicate defined over the pixels in set R_i and $\emptyset$ is the null set.

Cond. (a) indicates that the segmentation must be complete, or each pixel must be in a region while Cond. (b) suggests that regions must be disjoint with each other. Cond. (c) and Cond. (d) guarantee that all pixels in a segmented region R_i have the same properties, but different regions R_i and R_j are at least different in the sense of one predicate P.

Normally the term $\triangle_K(R) = \{R_1, R_2, \ldots, R_K\}$ is defined to denote the segment procedure with K denoting the number of the regions in $\triangle_K(R)$. In the proposed algorithm, an over-segmentation is performed on the image first of all to obtain an initial image partition $\triangle_{K_0}(R)$. It is assumed that there exists a sequence of region merges that transforms $\triangle_{K_0}(R)$ to the true partition $\triangle_{K^*}(R)$, here K^* is the number of the regions in $\triangle_{K^*}(R)$ and $K_0 \geq K^*$. This can be regarded as that each Region $R_i^{K^*}$ in $\triangle_{K^*}(R)$ is a union of certain regions in $\triangle_{K_0}(R)$. To get the sequence, a novel region merging method using a *k*-NN graph is applied on the initial partitions $\triangle_{K_0}(R)$. At each step of the merging process, the most similar pair of regions is merged and finally the true partitions $\triangle_{K^*}(R)$ is obtained.

Fig. 2 shows the flow of the proposed segmentation algorithm. And Fig. 1(a) from [14], which is piecewise constant image with white Gaussion noise, is taken as an example to show the algorithm visually. The aim of the first stage is to prepare for the following processing. In this stage an edge detection process is applied and other preprocessing can also be performed if needed. For example, Fig. 1(a) is filtered in practice to get the smooth image before further process. Fig. 1(b) shows the edge image of Fig. 1(a), and the use of the edge image will be explained in the next section. The second stege oversegments the filtered image to get the primitive partitions as described above. Watershed transform is adopted To Fig. 1 and the result is shown in Fig. 1(c). However, region-based algorithm may also be used to achieve oversegmentation. Both the two algorithms will be described in section IV. In the third stage *k*-NN graph is built based on the output of the initial partitions obtained in the second stage. In this stage a new region similarity measure function using local features along region edges is also computed as being introduced in section III. The implementation of *k*-NN graph construction is demonstrated in section IV. The final stage is region merging process using the *k*-NN graph. Two keys about the merging rule and stopping rule in this stage are discussed in the next section. Fig. 1(d) shows the final result of the proposed segmentation algorithm. Totally 7 regions are segmented in this process.

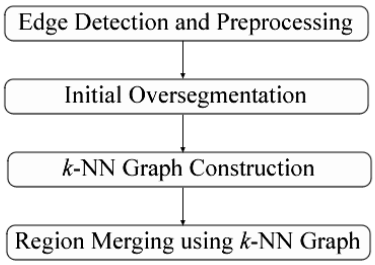


Fig. 2. Flow chart of the segmentation algorithm

III. Fast Region Merging using *k*-NN Graph

Our drive is to assign similar pixels in the same region, and get our expected segmentation by region merging. To achieve this, the definition of similarity between pixels and regions, the way to merge the region and the stopping rule to terminate the region merging process are the most important points. Therefore, in this section we will discuss all of this and demonstrate some of our ideas.

A. Region Similarity

Normally the difference of two regions' feature is computed to measure the similarity of them. For ease, global features are often extracted. i.e. the mean value of the pixels in a region R_i, spacial distance of two regions' centroid can be used to achieve this goal [15]. But the global feature may bring false merge. For example, if two big regions have a sharp difference along their edge while their global intensity means are almost the same, the algorithm will usually pick the two to merge. To overcome the drawback of global features, we propose a new region similarity based on the pixel's similarity.

We take a brightness image for example. For pixel p_i and p_j in image I, we define their similarity as:

$$\omega_{ij} = \begin{cases} e^{-\frac{\|I_i - I_j\|^2}{\sigma_1^2} - \frac{Edge^2(i,j)}{\sigma_2^2}} & \|X_i - X_j\| \le r \\ 0 & \|X_i - X_j\| > r \end{cases} \quad (2)$$

where X_i and I_i denote the coordinate and intensity of p_i respectively, $Edge(i,j)$ is the maximum value on the line connecting p_i and p_j in the edge image, which denotes the probability of an edge exists between p_i and p_j. Its definition is demonstrated in the next section. σ_1 and σ_2 is the parameter to modify the force of intensity and edge features in ω_{ij}. We also set a radius parameter r. If two pixels are too far away, or their distance is more than r, we directly set $\omega_{ij} = 0$. Here we just use intensity and edge feature, and use spatial distance to control the effective neighbors' range of p_i. If we want to use other features, we only need to define a function in the form like Edge feature and make ω_{ij} multiply with the defined item.

Let $d_i = \sum_j \omega_{ij}$ be the total connection from p_i to all other pixels. With the pixel similarity ω_{ij} and d_i, the similarity between region A and B is defined as:

$$W(A,B) = \frac{\sum_{i \in A, j \in B} \omega_{ij}}{\sqrt{(\sum_{i \in A} d_i)(\sum_{i \in B} d_i)}} \quad (3)$$

The region similarity is the sum of the pixel similarity between pixels from region A and B. To avoid the preference of merge between big regions, we make the sum divided by the normalized item, square root of the product of $\sum_{i \in A} d_i$ and $\sum_{i \in A} d_i$. $\sum_{i \in A} d_i$ and $\sum_{i \in A} d_i$ can be regarded as the volume of region A and B. Different from other definition, disjoint regions may have high similarity value in our definition. This can improve the detail parts, especially the small disjoint part of the segmentation. Besides, we can control the influence range of the region according to modification of the pixel similarity radius r. If r is small, similarity between two regions can be decided mainly by a part of pixels along their edge. According to the above formulation, the most similar pair of regions is the ones which have the high value of (3).

B. Merging using k*-nearest neighbor graph*

The traditional data structure for representing partitions is the region adjacency graph (RAG) [16]. The RAG of K-partition is defined as an undirected graph, and each graph node represent a region. If regions R_i and R_j are adjacent, a corresponding edge exists between node i and j.

There is no limit that edges must exist between adjacent regions in our region similarity definition (3), so every region may have more neighbors. As a result we choose a *k*-nearest neighbor (*k*-NN) graph proposed in [13] instead of RAG. *k*-NN graph is a weighted directed graph $G = (V, E, W)$. Same as RAG, V is the set of nodes representing regions and E is the set of edges representing pointers from a region to its neighbor regions. Every node has exactly k edges to the k nearest regions. All the regions similarities are computed and assigned to the corresponding edges as weight. In RAG, the search for the most similar pair of regions is repeated several times per iteration and every search requires $O(N)$ region similarity computation. The graph is utilized so that the search is limited only to the regions that are directly connected by the graph structure. This reduces the time complexity of every search from $O(N)$ to $O(k)$. The parameter k affects the quality of the final segmentation results and the running time. If the number of neighbors k is small, significant speedup can be obtained. And Pasi in [13] has proven that a small k can get a good approximated result.

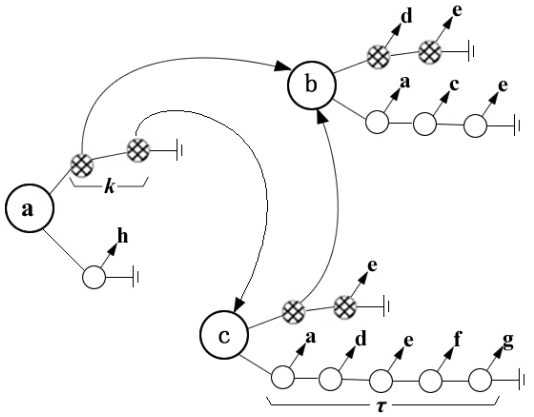


Fig. 3. Illustration of the double linked list structure for *k*-NN graph node ($k = 2$).

We use the double linked algorithm of *k*-NN graph to implement the merging process. Fig.3. shows the node structure of *k*-NN graph. For each node R_a, we maintain two lists: the *k*-NN list containing the pointers to its k nearest neighbors and another list containing the back pointers which point to the regions taking R_a as one of their k nearest neighbors. For example, in Fig.3, there are five regions that take region *c* as their nearest neighbors. All of them appear in the back pointer lists of *c(a,d,e,f,g)*. Using back pointers is to accelerate the process of searching the nodes whose nearest neighbor is the current one in merging process. We store the *k* nearest neighbors in descendent order so that the nearest neighbor is always the first one in the list. All nodes are stored in a heap by their similarity with the nearest one neighbor.

Given the *k*-NN graph of the initial *K*-partition, the merging process is instructed in the following algorithm.

Input: *k*-NN graph of *K*-partition
Iteration: For $i = 0$ to $n - 1$
Find the most similar pair (R_a, R_b) to be merged.
Merge the pair $(R_a, R_b) \rightarrow R_{ab}$.
Update the *k*-NN graph to $(K - i - 1)$ partition.
Output: *k*-NN graph of $(K - n)$-partition

In each merging iteration, finding the most similar pair of nodes takes constant time while the merging and updating process is more time consuming. After that, the node R_a and R_b are merged into one node R_{ab}. We select the *k* nearest neigh-

bors from the *2k* neighbors of the previously merged nodes R_a and R_b to keep the computation complexity reasonable. This means that the accuracy of the *k*-NN graph is compromised and, thus, the graph becomes an approximated nearest neighbor graph. It may also happen that the number of neighbors for the cluster R_{ab} can become smaller than *k*. At last the node R_a is replaced by $R_(ab)$ and the second node R_b is removed from the *k*-NN graph. We recompute the similarity with the neighbors of R_{ab}, which is a double process. In other words, we need compute both the edges from R_{ab} and the edges pointed to R_{ab}. At the same time, insertion sort is applied and no more than k nearest neighbors are kept. With the back pointer link, this procedure requires $O(\tau+\tau/k\cdot\log_2(\|\triangle_K(R)\|))$ [13], here τ denotes the number of nearest neighbor updates required at each iteration. Another action in graph updating is to update the heap. Deleting a node and modifying a node's position both need $O(\log_2(\|\triangle_K(R)\|))$ time. The merging step iterates no more than K times, so the merging procedure totally requires $O(\tau K\log_2(K))$ time.

C. Stop Condition

Predefining a K^* is the simplest way to stop the merging iteration. As long as the number of regions is K^*, the iteration stops automatically. But this need interaction and different images may need different K^*. Another way to stop the iteration is using the region similarity. If the global maximum region similarity value (3) is smaller than a certain threshold, the merging process will be terminated. This threshold can be set directly by user or be determined automatically by using the knowledge on the noise distribution (hypothesis testing)[17].

IV. Initialization and Graph Construction

In this section, we mainly discuss the initialization and implementation of our algorithm.

A. Initial Oversegmentation

Oversegmentation is the foundation of our algorithm. In our proposal, there are two requirements to the oversegmentation algorithm: (a) it must be simple, in other word, it must be implemented simply and get results quickly; (b) the primitive regions obtained should be appropriate, that is to say, the number of the primitive regions should be in a certain range, the size of regions should be appropriate, and the property in a region should be consistent which satisfies Cond. (c) in (1). Either one of watersheds or region growing method can be used under this consideration.

1) Watersheds-based Segmentation: The watershed algorithm proposed by Gauch [18] is a morphological method which overcomes the problem of disconnected contours and false edges. In the proposed algorithm, the fast watershed detection algorithm proposed by Vincent and Soille [12] is adopted. Let I be a digital brightness image. Watersheds are defined as the lines separating the so-called catchment basins with different minima of the image intensity. The catchment basin $C(M)$ is a set of pixels, which satisfies that, all water falling at the region inside $C(M)$, will flow down to the minimum M. The watersheds transform algorithm used here is based on immersion simulations [12], that is, on the recursive detection and fast labelling of the different catchment basins use queues. The algorithm consists of two steps: sorting and flooding. At the first step, the image pixels are sorted in ascendent order according to their intensities. With the image intensity histogram, a hash table is allocated in memory, where the i-th entry points to a list containing the image locations of intensity i. After that, the hash table is filled by scanning the image. Therefore, sorting scans the image twice while uses only constant memory. In the flooding step, the pixels are quickly accessed in ascendent intensity order (immersion) using the sorted image and labels are assigned to catchment basins. The label propagation is based on queues constructed using neighborhoods [12]. The output of the watershed algorithm is one-pixel wide watersheds and catchment basins (regions), so the watersheds line must be assigned to corresponding regions to obtain the real primitive partitions.

2) Region Growing: As its name implies, region growing is a procedure that groups pixels or subregions into larger regions based on predefined criteria. We use a simple algorithm in our paper. Same to watersheds, first we do a sorting on the image pixel by their intensity. Our algorithm will use a stack and predefine a threshold, namely color deviation σ. The algorithm starts with the smallest pixel, and set this pixel is the seed of a new region, then search its 8-neighbors, if the difference between the region and the neighbor's intensity is smaller than σ, label this pixel in the current region, push it in the stack too, and update the regions mean intensity. After all the pixel's neighbors are compared, pop a new pixel from the stack and repeat the process. If the stack is empty, search the next smallest pixel which is not assigned to any exist regions and create a new region. After all the pixels are assigned to certain regions, the algorithm is terminated. Because the each pixel is searched once, so the complexity of the algorithm is $O(N)$. And we can control the results according to modifying σ.

B. Edge Detection

Before creating the *k*-NN graph, an edge image E is drawn using the method proposed in [19]. Kirschsmask (4) and different rotations of it are applied to the brightness image, and the raw edge image is thresholded to obtain the final edge image E.

$$\begin{pmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{pmatrix} \tag{4}$$

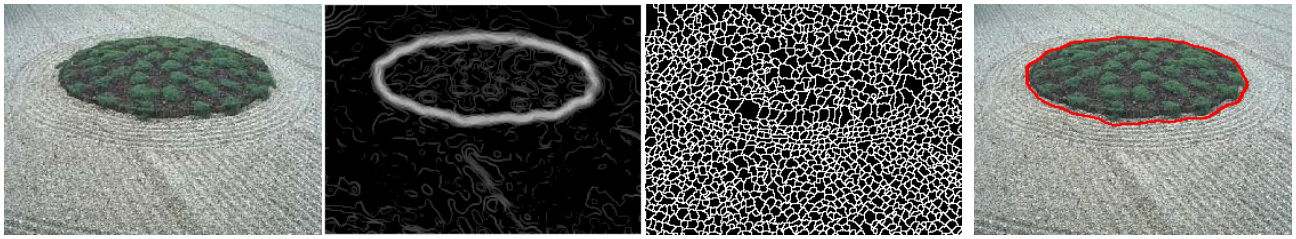
The edge response $Edeg(i,j)$ between pixel p_i and p_j in (2) is defined on the edge image E as below:

$$Edge(i,j) = \max_{k\in line(i,j)}(E_k) \tag{5}$$

where $line(i,j)$ is the line connected p_i and p_j. (5) actually denotes the maximum value on $line(i,j)$ in E.

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

Fig. 4. Segmentation of color homogeneous image. (a) Original image. (b) Edges detected in (a). (c) Primitive regions with region growing method ($\sigma = 8$, 97 regions). (d) Final segmentation ($\sigma_1 = 1$, $\sigma_2 = 0.1$).



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

Fig. 5. Segmentation of image with texture. (a) Original image. (b) Edges detected in (a). (c) Primitive regions with watershed transform (156 regions). (d) Final segmentation ($\sigma_1 = 1$, $\sigma_2 = 0.5$).

C. Creation of k-NN Graph

The pixels similarities are computed first and stored in an array W for the use of computing region similarity. Similarly d_i in (3) are pre-computed and stored in array D. After oversegmentation, in the procedure of assigning pixels to regions, we add d_i to the corresponding region to compute its volume. All of this require $O(r^2N)$ time with N denoting the number of the image, and r being the pixel similarity radius.

We use *brute force* to compute the region similarity $W(a, b)$. So it is more time consuming. Let $\triangle_{K_0}(R)$ be the primitive segmentation. For a region R_a We define an array S of size K_0 to contain the similarity with other regions. All the values in S are set to 0. We travel every pixel in R_a. If the pixel has a neighbor in region R_b, we add the corresponding pixel similarity to $S[b]$. Then we pick up all the value $S[b]$ more then 0 and divide them by the volume of R_a and R_b to obtain the $W(a, b)$. We use insert sort to add R_b to the nearest neighbor link of R_a. All of this need K_0^2 travels, fortunately the similarity definition is symmetric, so we can save half of the time. After all the neighbors are computed, only k nearest neighbors are kept in every node. At the same time, the back pointer link is constructed. The whole procedure's complexity is $O(K_0^2)$. Finally apply a heap sorting on the k-NN graph with the complexity of $O(K_0 \log_2(K_0))$.

V. Experimental Results and Discussion

The images(130 $\times$ 130, 8 b/pixel) shown in Fig. 4 and images(278 $\times$ 202, 8 b/pixel) shown in Fig. 5 are used in order to illustrate the stages of the segmentation algorithm and visually assess the quality of the segmentation results. From left to right, Fig. 4(a) and Fig. 5(a) are the inputs. Fig. 4(a) is color homogeneous with high contrast fore- and back-ground. But the right top part in background has a brown region the same as the color of the boy's hair, which could disturb the segmentation if only color feature is used. Fig. 5(a) has the noise of little texture in the road and flower bed.

At the beginning, edge detection is performed on the two source images and the obtained edge images are shown in column (b) of Fig. 4 and Fig. 5 respectively. Different initial oversegmentation algorithms are used to compare the rusults. To Fig. 4, we adopt region growing method directly since the color is homogeneous. We set color deviation $\sigma = 8$ and get 97 regions after the oversegmentation. Fig. 4(c) shows the oversegmentation results. To reduce the noise's influence, watershed transform is performed on Fig. 5 just like Fig. 1. Fig. 5(c) shows the watershed detected in Fig. 5(a). Totally 156 regions are partitioned in this process.

The initial partitions are used for the construction of k-NN graph, and the merging process begins. The number of regions obtained in initial oversegmentation determines the computational and memory requirements for the construction and processing(merging) of the k-NN. In the process of computing the pixel similarity, we set $\sigma_1 = 1$, $\sigma_2 = 0.1$ for Fig. 4 and $\sigma_1 = 1$, $\sigma_2 = 0.5$ for Fig. 5 respectively since Fig. 5 has noise of texture. The final segmentation results are shown in Fig. 4(d) and Fig. 5(d). The boy and the garden are both segmented from the background.

Fig. 6 shows two application examples. Fig. 6(a) is a natural grayscale image (250 $\times$ 250, 8 b/pixel) from Matlab's image database, namely cameraman. We use region growing method

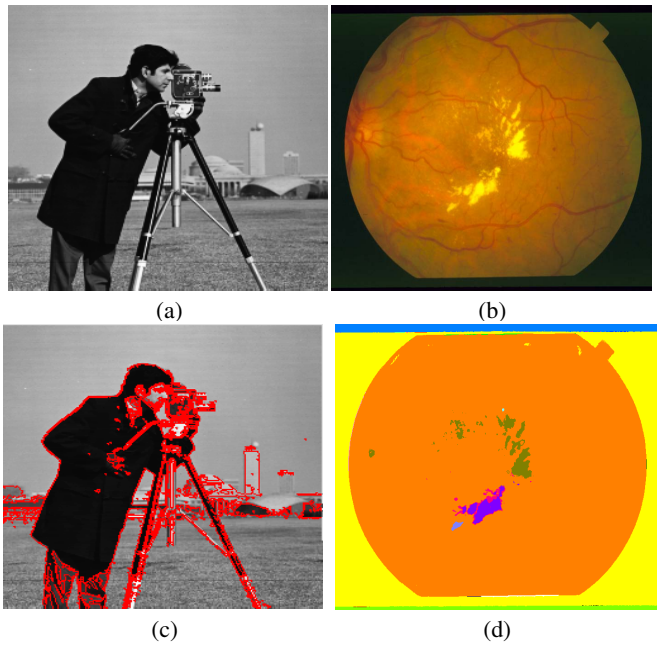


Fig. 6. Two application examples. (a) A natural image. (b) A retinal image. (c) Segmentation result of (a) (19 regions). (d) Segmentation result of (c) (10 regions).

to get the initial partitions. In our computer (CPU: AMD athlon 64bit 3000+, Memory: 2G), it takes 17 seconds to obtain the final results. We get 19 regions and show the regions contour in Fig. 6(c). The main background was segmented whereas some building was also extracted from background. Fig. 6(b) is a colorful retinal image (700 $\times$ 605, 24 b/pixel) for detecting the hard exudates of diabetes patients. The hard exudates (small yellowwhite patches in the middle of the image) normally have no regular shape, and are distributed like liquid. Normal segmentation algorithm often cannot segment them. However, with special enhancement [19], our algorithm can extract the hard exudates exactly in Fig. 6(d) due to our local region similarity definition. In Fig. 6(d), some of the regions are even disjoint. This proves that, our algorithm can improve the segmentation detail as our expectation.

VI. Conclusion

A fast image segmentation method using region merging is presented which integrates color and edge features in a unit framework via the oversegmentation algorithm. The algorithm can handle colorful or grayscale image and obtain the output of the partition regions of the image. It can be the input of many further image processing tasks. The output of our algorithm can also be one-pixel wide contours/surfaces according to a fast extraction based on the partition regions. In our proposal, the new region similarity definition based on local pixel similarity can use kinds of image features in a unit form. Regions are merged according to the pixels similarity along their edge instead of the global mean feature distance. In this way, the drive of assigning similar pixels in the same region can be actually realized. Furthermore, a k-NN graph is used to accelerate the merging process and kinds of stop conditions can be used to determine the termination of the merge iteration.

We implement the algorithm and test it with kinds of cases. The satisfactory results prove our method is robust and computation efficient, the segmentation performance is also encouraging. Nevertheless, the memory requirements are relatively high due to storing the pixels similarity. Besides, our algorithm has many parameters, this needs user to set the parameters manually in partitioning different images or even test them with interaction. In our current version, we only use the color and edge features while there are still many other features can be used in our segmentation framework, such as gratitude, spacial distance and texture. So next we will try to include more features. The future work will also be focused on automatical parameter determination and merge process acceleration. Of cause, the 3-D version of the algorithm is also under consideration.

References

[1] P. Suetens, P. Fua, and A. J. Hanson, *"Computational strategies for object recognition,"* ACM Comput. Surv., vol. 24, pp. 5C61, Mar. 1992.

[2] P. Besl and R. Jain, *"Three-dimensional object recognition,"* ACM Comput. Surv., vol. 17, pp. 75-145, Mar. 1985.

[3] K. Hohne, H. Fuchs, and S. Pizer, *3D Imaging in Medicine: Algorithms, Systems, Applications*. Berlin, Germany: Springer-Verlag, 1990.

[4] M. Bomans, K. Hohne, U. Tiede, and M. Riemer, *"3-D segmentation of MR images of the head for 3-D display,"* IEEE Trans. Med. Imag., vol. 9, pp. 253-277, June 1990.

[5] M. Kunt, M. Benard, and R. Leonardi, *"Recent results in highcompression image coding,"* IEEE Trans. Circuits Syst., vol. 34, pp. 1306-1336, Nov. 1987.

[6] P. Willemin, T. Reed, and M. Kunt, *"Image sequence coding by split and merge,"* IEEE Trans.Commun., vol. 39, pp. 1845-1855, Dec. 1991.

[7] N. Pal and S. Pal, *"A review on image segmentation techniques,"* Pattern Recognit., vol. 26, pp. 1277-1294, 1993.

[8] M. G. Liu, J. Jiang and C.H. Hou, *"Hybrid Image Segmentation Using Watersheds and Adaptive Region Growing,"* Visual Information Engineering, 7-9 July 2003, pp. 282-285, 2003.

[9] R. Beveridge et al., *"Segmenting images using localized histograms and region merging,"* Comput. Vis., Graph., Image Process., vol. 2, pp. 311-347, 1989.

[10] Z. Wu, *"Homogeneity testing for unlabeled data: A performance evaluation,"* CVGIP: Graph. Models Image Process., vol. 55, pp. 370-380, Sept. 1993.

[11] F. Meyer and S. Beucher, *"Morphological segmentation,"* J. Vis. Commun. Image Represent., vol. 1, pp. 21-46, Sept. 1990.

[12] L. Vincent and P. Soille, *"Watersheds in digital spaces: An efficient algorithm based on immersion simulations,"* IEEE Trans. Pattern Anal. Machine Intell., vol. 13, pp. 583-598, June 1991.

[13] P. Franti, O. Virmajoki and V. Hautamaki, *"Fast Agglomerative Clustering Using a k-Nearest Neighbor Graph,"* IEEE Trans. Pattern Anal. Machine Intell., vol. 28, pp. 1875-1881, Nov. 2006.

[14] K. Haris, S. N. Efstratiadis and K. Katsaggelos, *"RHybrid Image Segmentation Using Watersheds and Fast Region Merging,"* IEEE Trans. Img. Proc., vol. 7, no. 12, Dec. 1998.

[15] F. Abad, J. Garcia-Consuegra and G. Cisneros, *"Merging regions based on the VDM distance,"* in Proc. IGARSS, vol.2, pp. 615-617, July 2000.

[16] X. Wu, *"Adaptive split-and-merge segmentation based on piecewise least-square approximation,"* IEEE Trans. Pattern Anal. Machine Intell., vol. 15, pp. 808-815, Aug. 1993.

[17] K. Haris, *"A hybrid algorithm for the segmentation of 2D and 3D images,"* Masters thesis, Univ. Crete, 1994.

[18] J. M. Gauch, *"Image segmentation and analysis via multiscale gradient watershed hierarchies,"* IEEE Trans. Img. Proc., vol. 8, no. I, Jan. 1999.

[19] C. I. Sanchez, R. Hornero, M. I. Lopez and J. Poza, *"Retinal Image Analysis to Detect and Quantify Lesions Associated with Diabetic Retinopathy,"* in Proc. of the 26^{th} Annual International Conference of the IEEE EMBS, San Francisco, CA, USA, pp. 1624-1627, 2004.