Annales UMCS Informatica AI 5 (2006) 191-200

Annales UMCS
Informatica
Lublin-Polonia
Sectio AI

http://www.annales.umcs.lublin.pl/

Multilevel near optimal thresholding applied to watershed grouping

Jakub Smołka*

Institute of Computer Science, Faculty of Electrical Engineering and Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

The major drawback of watershed transformation is over-segmentation. It also has a significant advantage: very good edge extraction. Thresholding methods usually utilize only global information such as an image histogram; however, they have the ability to group pixels into clusters by their value. The method presented in this paper combines the advantages of watershed segmentation and multilevel thresholding. This was achieved by modifying selected optimal thresholding methods so that they treat watersheds as a whole and using those methods in a multilevel thresholding algorithm for grouping watersheds. Otsu's, Kapur's, maximum entropy and step function approximation thresholding methods have been tested. The obtained results are presented and discussed.

1. Introduction

Watershed transformation is a method that simulates pouring water onto a landscape created on the basis of a digital image. Unfortunately transformation produces a region for each local minimum so, usually, the number of watersheds (catchment basins) is too big. This is the major drawback of watershed transformation and is called over-segmentation. The watershed transformation has also a significant advantage: very good edge extraction. Multilevel thresholding, on the other hand, does not take into account spatial information. It utilizes only global information such as an image histogram. However thresholding has the ability to group pixels into clusters by their value. The method presented in this paper combines the advantages of watershed segmentation and multilevel thresholding. It generates an over-segmented image and then groups watersheds (catchment basins) on the basis of their mean value. No catchment basin is divided; hence, multilevel thresholding takes into account: boundaries determined by watershed transformation.

* *E-mail address*: jakub.smolka@pollub.pl

2. Watershed transformation

Watershed transformation (also called watershed segmentation) was introduced by Beucher and Lantuejoul in [1]. Its principle is very straightforward and can be easily explained by analogy to rain pouring on to a landscape. When water rains on to a landscape, it flows with gravity to collect in low catchment basins. The size of those basins grows as the amount of precipitation they receive increases. At a certain point basins would start spilling into one another, causing small basins to merge together into larger basins. To prevent them from merging, the transformation algorithm starts to build dams between them.

As mentioned above watershed transformation treats an image $I(x)$ as a height function that describes a landscape. It assumes that higher pixel values indicate the presence of boundaries in the original image $f(x)$. That is why the gradient operator is often used in the watershed transformation for obtaining the height function $I(x) = |\nabla f(x)|\nabla$. This is illustrated in figure 1.

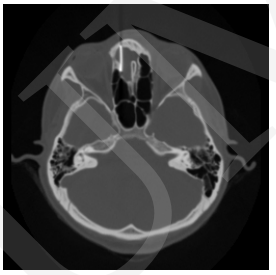
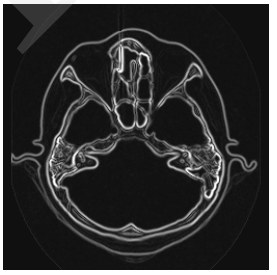
a)  b) 

Fig. 1. Obtaining the height function I, (a) original image, (b) height function

Since the gradient operator is very sensitive to noise, the original image is usually filtered with an edge preserving smoothing, filter such as gradient diffusion [2]. Despite the smoothing, the height function $I(x)$ usually still has many local minima. Additionally there are as many catchment basins in an image as there are local minima in $I(x)$. Here is where the major drawback of watershed segmentation appears – it produces too many regions and over-segmentation results [3,4]. Over-segmentation is illustrated in figure 2.

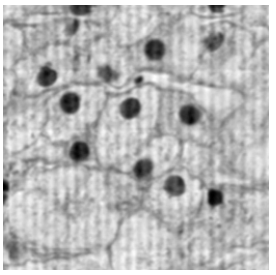
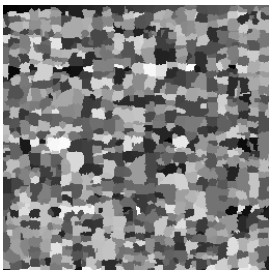
a)  b) 

Fig. 2. Over-segmentation, (a) original image, (b) watersheds in an over-segmented image

To alleviate this problem, one can establish a minimum watershed depth or threshold the gradient image $I(x)$ prior to segmentation, using a small threshold [3]. These solutions, however, are very simple and usually do not give satisfactory results.

3. Optimal watershed thresholding

In order to enable multilevel thresholding to group watersheds, optimal thresholding must be adapted for the use with images divided into catchment basins. It is similar to the pixel version of optimal thresholding. The difference is that the thresholding is performed on the basis of watershed mean values instead of pixel values. As a consequence, one of the mean values is the optimal threshold. The catchment basins, whose mean value is lower than the threshold, belong to the background and the others belong to the segmented object. The algorithm treats catchment basins as a whole. They may either be included in an object or in the background and cannot be divided into parts. Individual pixels within a watershed may have values above or below the threshold mean. The optimal thresholding function depends on two parameters, M_{min} and M_{max}, which are the minimal and maximal allowed catchment basin means. Watersheds with mean values that fall outside this range are ignored. At the beginning, the brightest (with the highest mean) watershed is added to the bright object (and, as a consequence, taken from dark background). Multiple catchment basins are added if more than one has the same mean value. Each time watersheds are added, value of the assessment function is calculated. Then, in a loop, watersheds with consecutive mean values are added to the object, and the assessment function is calculated. If a better threshold is found, it is saved as the optimal threshold M_{opt}. The algorithm can use many different assessment functions, including those described below. In this paper the rule, that the higher the function value the better segmentation, is used. Below is the pseudo-code description of this algorithm:

```
O_tmp ← ∅                                          //find watershed with the highest
M_tmp ← findHighestMeanLowerThan(M_max);   //allowed mean and add it to the
O_tmp ← O_tmp ∪ findWatershedsWithMean(M_tmp);                    //object
F_tmp ← assessmentFunction(O_tmp);          //calculate assessment function
F_opt ← F_tmp;                                                  //for current object
M_opt ← M_tmp;
while M_tmp ≥ M_min do                      //start looking for optimal threshold
   M_tmp ← findHighestMeanLowerThan(M_tmp);
   O_tmp ← O_tmp ∪ findWatershedsWithMean(M_tmp);
   F_tmp ← assessmentFunction(O_tmp);
   if F_tmp > F_opt then                                  //if current assessment function
      M_opt ← M_tmp;                                                  //value is higher
```

```
    F_opt ← F_tmp;                    //save new optimal function and mean values
  end if;
end while;
Result ← M_opt
```

where: M_{tmp} – mean of a watershed, F_{tmp} – value of assessment function, F_{opt} – value of assessment function corresponding to optimal mean, M_{opt} – optimal mean value that divides allowed mean range, O_{opt} – set of watersheds that constitute object, $[M_{min}, M_{max})$ – allowed mean range.

4. Multilevel thresholding

The multilevel thresholding method is a modified version of the fast iterative scheme for multilevel thresholding methods by Peng-Yeng Yin and Ling Hwei Chen [5]. It was adapted for clustering watersheds. Other modifications are also introduced. The algorithm finds multiple near optimal thresholds using the optimal watershed thresholding method described above. Thresholds are kept in a *ranges* array which, initially holds only two thresholds: the lowest watershed mean and infinity. This allows for some algorithm simplification and also means that, at the beginning, there is only one class. After initialization the algorithm starts dividing the image into the requested (by user) number of classes. First it looks for the mean range to be divided by determining which class has the highest D_k parameter. This parameter is the absolute value of the difference between the number of mean value occurrences and the histogram's maximum.

$$D_k = \left| H_k\left[ceil\left(M_k\right)\right] - \max_{i=0\ldots n-1} H_k\left[i\right] \right|,$$

where: k – class (range) number, H_k – *k-th* class histogram, M_k – *k-th* mean class value, $ceil(M_k)$ – the smallest integer number greater than *k-th* class mean value, n – number of gray levels.

When a range to be divided is found, optimal thresholding is performed but it is limited to this watershed mean range. After a new threshold has been found, it is inserted into the ranges array. Then the thresholds are copied into a temporary array and are optimized. Threshold optimization consists in taking into consideration consecutive threshold triples and replacing the middle one with a new threshold found by using the optimal thresholding method. If the assessment function value increaseses, the new thresholds are saved and the optimization continues. Otherwise, the algorithm proceeds to adding a new threshold. The process stops when the requested number of thresholds is found. Below it the pseudo-code description of the described algorithm:

```
ranges[0] ← findLowestMean();
ranges[1] ← ∞;
n_t ← 2;                                   //there are two thresholds
for i ← 0 to n_c-1 do
```

```
//new threshold addition
r_DkMax ← findRangeWithMaxDk();      //returns n for [ranges[n],ranges[n+1])
M_new ← optimalThresholding(r_DkMax,r_DkMax+1);
insertMeanAt(r_DkMax+1,M_new,ranges);          //inserts M_new between
n_t ← n_t+1;                          //ranges[r_DkMax] and ranges[r_DkMax+1]

//threshold optimalization
Q_b ← assessThresholds(ranges);
repeat
   tempRanges ← ranges;                                  //copy all thresholds
   for j ← 1 to n_t-2 do                              //try to find better thresholds
      tempRanges[j] ←                                    //within consecutive ranges
         optimalThresholding(tempRanges[j-1],tempRanges[j+1]);
   end for;
   Q_t ← assessThresholds(tempRanges);
   if Q_t > Q_b then                                        //save better thresholds
      Q_b ← Q_t;
      ranges ← tempRanges;
   end if;
until Q_t < Q_b;                          //the thresholds couldn't be improved
end for;
Result ← ranges;
```

where: n_c – requested number of classes, n_t – current number of thresholds, ranges, Q_b, Q_t – best/temporary assessment function value, *ranges* array containing thresholds.

5. Assessment functions

The watershed optimal thresholding algorithm can use many different assessment functions. Four selected methods are presented here. They are region based methods, meaning that they take into account entire classes and not only their boundaries [6]. Additionally, all the functions still operate on separate pixels. Watersheds only determine which pixels are taken into consideration. In the equations below the following symbols are used: c – number of classes, n – number of gray levels in image, N – number of points in image, $p_k(i)$ – probability of selecting pixel of gray level i, belonging to *k-th* class, $H_k(i)$ – number of pixels of gray level i, belonging to *k-th* class, R – segmented image, x – a pixel, $I(x)$ – value of x, R_k – *k-th* class, $f(R)$ – assessment function, $\overline{I}_k$ – average pixel value in *k-th* class, σ_k – standard deviation of *k-th* class.

First, there is the maximum entropy method commonly used for pixel based optimal thresholding [7-9]. The method consists in maximizing the sum of entropies of classes into which the image is divided. If $p_k(i)$ is the probability of

selecting a pixel of gray level i belonging to *k-th* class, then the entropy for c classes is given by:

$$f(R) = -\sum_{k=1}^{c} \sum_{i=1}^{n-1} p_k(i) \cdot \ln p_k(i), \quad p_k(i) = H_k(i)/N .$$

The equation above is used by multilevel thresholding. For the use of (two class) optimal thresholding, it can be rewritten for two classes:

$$f(R) = -\sum_{i=0}^{n-1} p_1(i) \cdot \ln p_1(i) - \sum_{i=1}^{n-1} p_2(i) \cdot \ln p_2(i),$$
$$p_1(i) = H_1(i)/N, \quad p_2(i) = H_2(i)/N.$$

Second, Kapur's method is also an entropy based method [5]. It differs from the maximum entropy method in the way it calculates probability. Instead of $p_k(i)$ it takes $p_k(i)/\omega_k$. This is the probability of selecting a pixel of gray level i in *k-th* class, calculated with respect to the number of pixels in that class and not with respect to the number of pixels in the entire image. If c is the number of classes in the image, then the Kapur's assessment function is given by:

$$f(R) = -\sum_{k=1}^{c} \sum_{i=0}^{n-1} \frac{p_k(i)}{\omega_k} \cdot \ln \frac{p_k(i)}{\omega_k},$$
$$\omega_k = \sum_{i=0}^{n-1} p_k(i),$$
$$p_k(i) = H_k(i)/N .$$

In the case of two class segmentation, it can be rewritten as:

$$f(R) = -\sum_{i=0}^{n-1} \frac{p_1(i)}{\omega_1} \cdot \ln \frac{p_1(i)}{\omega_1} - \sum_{i=0}^{n-1} \frac{p_2(i)}{\omega_2} \cdot \ln \frac{p_2(i)}{\omega_2},$$
$$\omega_1 = \sum_{i=0}^{n-1} p_1(i), \quad \omega_2 = \sum_{i=0}^{n-1} p_2(i)$$
$$p_1(i) = H_1(i)/N, \quad p_2(i) = H_2(i)/N.$$

The next optimal thresholding method is based on the distance to a gray level function [6]. In this method the classes are approximated by their mean value. Then the distance between the original image and the constructed function is calculated. The distance is inverted so that a better approximation (smaller distance) corresponds to a better segmentation. The distance inverse for c classes is given by:

$$f(R) = \frac{1}{\sqrt{\sum_{k=1}^{c} \sum_{x \in R_k} \left[I(x) - \bar{I}_k \right]^2}} .$$

For two classes the equation can be rewritten as:

$$f(R)=\frac{1}{\sqrt{\sum_{x\in R_1}\left[I(x)-\bar{I}_1\right]^2+\sum_{x\in R_2}\left[I(x)-\bar{I}_2\right]^2}}.$$

Otsu's method is the last optimal thresholding method presented in this paper. It finds a threshold that minimizes weighted within-class variance and simultaneously maximizes between-class variance since the total variance of the image is constant. Within-class variance for *c* classes is simpler to calculate than between-class variance; hence, the former is used in multi-class segmentation quality assessment. By inverting that value, it is possible to go from minimization to maximization. The formula being maximized by the algorithm is:

$$f(R)=\frac{1}{\sum_{k=1}^{c}\omega_k\cdot\sigma_k^2},$$

$$\omega_k=\sum_{i=0}^{n-1}p_k(i),$$

$$p_k(i)=H_k(i)/N.$$

The above equation can be rewritten for 2 classes:

$$f(R)=\frac{1}{\omega_1\cdot\sigma_1^2+\omega_2\cdot\sigma_2^2},$$

$$\omega_1=\sum_{i=0}^{n-1}p_1(i),\qquad \omega_2=\sum_{i=0}^{n-1}p_2(i)$$

$$p_1(i)=H_1(i)/N,\qquad p_2(i)=H_2(i)/N.$$

6. Results

The following figures show the results obtained with multilevel thresholding adapted for use with watersheds. Images with at least three visible classes have been selected. Each test image was preprocessed with an edge preserving gradient diffusion smoothing filter [2] in order to remove noise. Before applying the watershed transformation, the gradient image was calculated. The gradient image was in turn thresholded to reduce over segmentation (using threshold equal to 5% of the value range), and the watershed transformation was applied. The number of classes was arbitrarily selected. In the presented results it is equal either to 3 or 4. Figures 3 and 4 show original and thresholded MRI scans. The segmented picture of a liver tissue is in Figure 5. All of the presented assessment functions were tested.

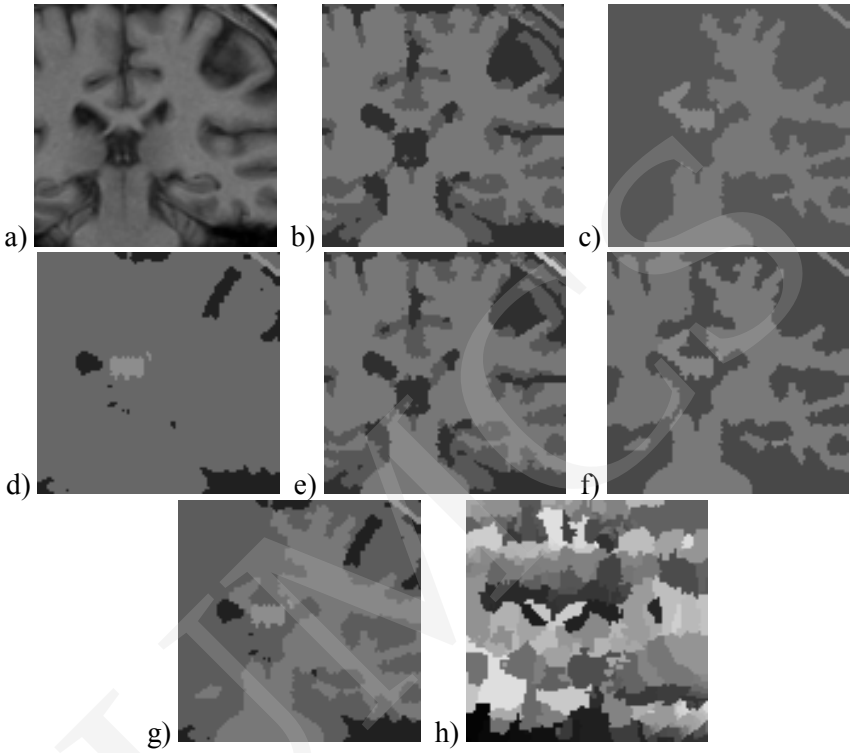


Fig. 3. MRI scan, (a) original image, results obtained using (b) Otsu's (3 classes), (c) entropy maximization (3 classes), (d) Kapur's (3 classes), (e) Otsu's (4 classes), (f) entropy maximization (4 classes) and (g) Kapur's assessment functions (4 classes), (h) watersheds before grouping (MR brain data set 788_6_max and its manual segmentation was provided by the Center for Morphometric Analysis at Massachusetts General Hospital and is available at http://neuro-www.mgh.harvard.edu/cma/ibsr.)

Entropy maximization and Kapur's method did not give satisfactory results. They had a tendency to select thresholds located close to one another and close to a lower boundary of the brightest class. This resulted in two large bright and dark classes along with one or two small mid-tone classes being segmented. Otsu's and step function methods on the other hand give very good results. Selected thresholds are located close to the local minima in the histogram, and segmented classes correspond well to objects a human viewer is able to distinguish in the original image. Both methods in most cases give nearly identical results. Additionally they are both easy to implement and to optimize for the use with images divided into watersheds, so the algorithm that utilizes them is noticeably faster.

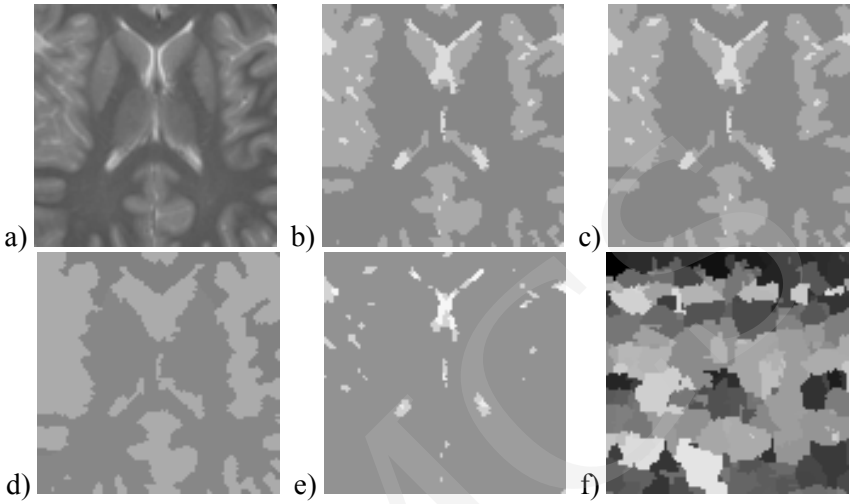


Fig. 4. MRI PD scan, (a) original image, results obtained using (b) Otsu's (3 classes), (c) step function approximation (3 classes), (d) entropy maximization (3 classes) and (e) Kapur's assessment functions (3 classes), (f) watersheds before grouping (m_vm10xx data set from the Visible Human Project was used)

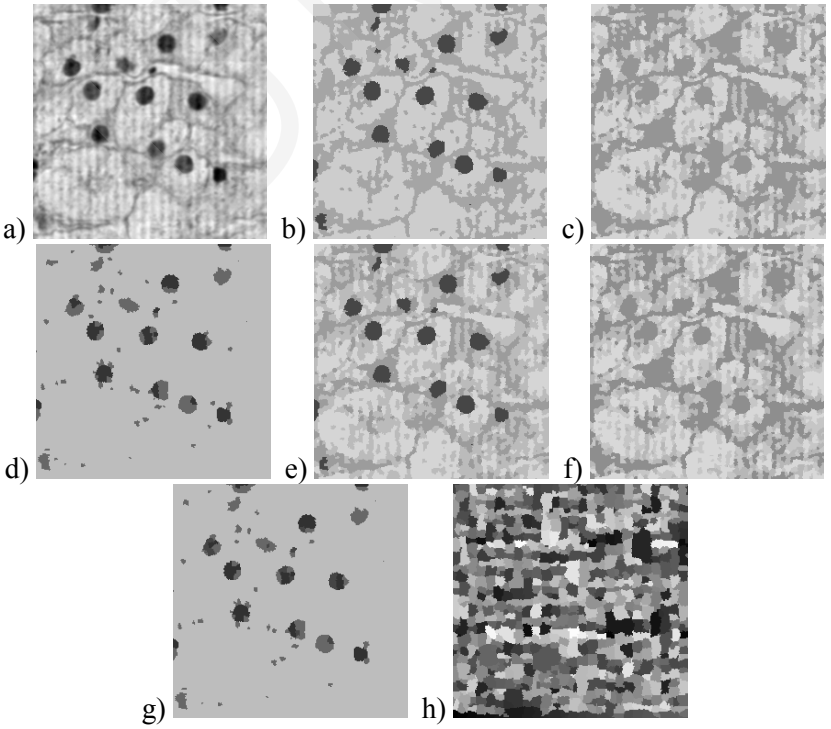


Fig. 5. Liver tissue, (a) original image, results obtained using (b) Otsu's (3 classes), (c) entropy maximization (3 classes), (d) Kapur' s (3 classes), (e) Otsu's (4 classes), (f) entropy maximization (4 classes) and (g) Kapur's assessment functions (4 classes), (h) watersheds

Conclusions

The results presented above show that multilevel thresholding can be used successfully for joining watersheds in an over-segmented image. Use of watersheds gives thresholding methods the ability to segment regions with boundaries corresponding to those of the object depicted in the original image. This allows for obtaining more accurate results. Multilevel thresholding requires a one-level optimal thresholding method to be adapted for the use with watersheds. The modification consists in using the watershed mean values instead of separate pixel values and not allowing the algorithm to divide catchment basins when searching for an optimal threshold. Four different methods have been appropriately modified including Otsu's, Kapur's, maximum entropy and step function approximation optimal thresholding algorithms. Only Otsu's and step function methods gave good results. They were also easier to implement and to optimize for the use with watersheds; as a consequence they are much faster.

Multilevel thresholding adapted for the use with watersheds may be also applied as a preprocessing algorithm for watershed region growing [10], which requires two-class images as an input. Once the thresholds are found, selecting pairs of classes is straightforward. The author intends to investigate this possibility in a future publication.

References

[1] Beucher S., Lantuejoul C., *Use of watersheds in contour detection*, International Workshop on Image Processing: Real-time Edge and Motion Detection/Estimation, (1979).
[2] Perona P., Malik J., *Scale-space and edge detection using anisotropic diffusion*, IEEE Transactions on Pattern Analysis Machine Intelligence, 12 (1990) 629.
[3] Ibanez L., Schroeder W., Ng L., Cates J., *The ITK Software Guide*, Kitware Inc., (2003).
[4] Beucher S., *The watershed transformation applied to image segmentation*, Scanning Microscopy International, 6 (1992) 299.
[5] Peng-Yeng Yin, Ling Hwei Chen, *A Fast Iterative Scheme for Multilevel Thresholding* Methods, Signal Processing, 60 (1997) 305.
[6] Revol-Muller Ch., Peyrin F., Carrillon Y., Odet Ch., *Automated 3D Region Growing Algorithm Based on an Assessment Function*, Pattern Recognition Letters, 23 (2002) 137.
[7] Gonzalez R.C., Woods R.E., *Digital Image Processing*, Addison-Wesley Publishing Company, (1993).
[8] Myler H.R., Weeks A. R., *The Pocket Handbook of Image Processing Algorithms in C*, Prentice Hall PTR .
[9] Seul M., O'Gorman L., Sammon M. J., *Practical Algorithms for Image Analysis: Description, Examples, and Code*, Cambridge Univerisity Press, (2000).
[10] Smołka J., *Watershed based region growing algorithm*, Annales Informatica UMCS, Lublin, 3 (2005) 169.