Otsu's Criterion-based Multilevel Thresholding by a Nature-inspired Metaheuristic called Galaxy-based Search Algorithm

Hamed Shah-Hosseini
Faculty of Electrical and Computer Engineering
Shahid Beheshti University
Tehran, Iran
e-mails: tasom2002@yahoo.com, h_shahhosseini@sbu.ac.ir

***Abstract*— In this paper, image segmentation of gray-level images is performed by multilevel thresholding. The optimal thresholds for this purpose are found by maximizing the between-class variance (the Otsu's criterion). The optimization is conducted by a newly-developed nature-inspired metaheuristic called "Galaxy-based Search Algorithm" or the GbSA. The proposed GbSA resembles the spiral arms of some galaxies to search for the optimal thresholds. The GbSA also uses a modified Hill Climbing algorithm as a local search. The experimental results show that the GbSA finds the optimal or very near optimal thresholds in all runs of the algorithm.**

***Keywords-:** Image segmentation; thresholding; metaheuristic; optimization; Otsu; chaos;*

I. INTRODUCTION

Image segmentation is often one of the main tasks in Image Processing and Computer Vision. Image segmentation is a process by which the whole image is segmented into several regions based on similarities and differences that exist between the pixels of the input image. Each region should contain an object of the image at hand. Therefore, by doing segmentation, the image is divided into several sub-images such that each sub-image represents an object of the scene.

Multilevel thresholding is among the techniques that can be used for image segmentation. For this purpose, the number of thresholds is given in advance. Then, the optimal thresholds are often found by maximizing or minimizing a criterion. One of the best criteria for thresholding was introduced by Otsu [1] in which the image is assumed to be composed of only two regions: object and background, and the best threshold is the one that maximizes the between-classes variance of the two regions. However, when the number of thresholds is increased, exhaustive search algorithms fail to find the optimal thresholds in a reasonable time. As a result, other search techniques which can deal with huge search spaces should be used.

Metaheuristics are favorable methods to deal with optimization problems whose search spaces for optimal solutions are extremely vast. For multilevel thresholding, metaheurisitcs such as Genetic Algorithms, Particle Swarm Optimization, Ant Colony Optimization, and Intelligent Water Drops have been introduced [2-5].

In this paper, a newly introduced metaheuristic algorithm called "Galaxy-based Search Algorithm" or GbSA is introduced [6] for multilevel thresholding. For this purpose, the Otsu's criterion function is maximized by the proposed GbSA method to obtain optimal thresholds. The GbSA searches the input space using a spiral chaotic movement. In fact, the GbSA approximates one arm of a spiral galaxy to search its environment. This spiral movement is enhanced by a chaotic process to help the search space exploration.

From now on, the proposed GbSA for multilevel thresholding is called "GbSA-MLT". The GbSA-MLT begins its work with some equally-spaced thresholds, which form the initial solution. Then, the GbSA moves spirally from the initial solution, which resembles the core of a galaxy. The galaxy's arm moves spirally to search the surrounding of the core until it finds a fitter solution. After that, a local search algorithm is activated from the newly-found solution to obtain the best local solution around. This local search algorithm may be chosen to be a hill-climbing search [7] algorithm or one of its modified versions. Here, a modified version of Hill Climbing is employed. The solution obtained by the local search is used as a new core for the GbSA, and the whole process is repeated again until a predetermined stopping condition(s) is satisfied.

The rest of the paper is organized as follows: Next section reviews multilevel thresholding. Section III introduces the proposed GbSA. Final section, section IV, contains the experimental results with some well-known images.

II. MULTILEVEL THRESHOLDING

Multilevel thresholding uses a number of thresholds $\{S_1, S_2, ..., S_L\}$ in the histogram of the image $f(x, y)$ to separate the pixels of the objects in the image. By using the obtained thresholds, the original image is thresholded and the segmented image $T(f(x, y))$ is created. Specifically:

$$T(f(x,y)) = \begin{cases} g_0 & \text{if } f(x,y) \le S_1 \\ g_1 & \text{if } S_1 < f(x,y) \le S_2 \\ \cdots & \cdots\cdots\cdots \\ g_L & \text{if } f(x,y) > S_L \end{cases} \quad (1)$$

Such that g_i is the gray-level assigned to all pixels of the region i, which eventually represents object i . As it is seen in (1), the $L+1$ regions are determined by the L thresholds $\{S_1, S_2, ..., S_L\}$. The value of g_i may be chosen to be the mean value of gray-levels of the region's pixels. However, in this paper we use the maximum range of gray-levels, 255, to distribute the gray-levels of regions equally. Specifically, $g_i = i.\left\lfloor \frac{255}{L} \right\rfloor$ such that the function $\lfloor . \rfloor$ returns the integer value of its argument.

III. THE PROPOSED GALAXY-BASED SEARCH ALGORITHM

It is mentioned that the proposed GbSA may be fallen into the category of variable neighborhood search algorithms. In a variable neighborhood search algorithm [8], a set of neighborhood structures is defined first. Then, the algorithm searches from the nearest neighborhood set to the farthest one until it finds a better solution. However, the exact structure of the neighborhood sets and the sizes of them are not available. Moreover, the number of neighborhood sets is another factor unknown to the user. Thus, the user has to define the neighborhood sets and their related parameters before using a variable neighborhood search algorithm. Here, the proposed GbSA uses a spiral-like movement in each dimension of the search space with the help of chaotic steps and constant rotation around the initial solution. Gradually, the arm of the galaxy opens and covers the search space in order to find a better solution. The pseudo-code of the GbSA is shown in Fig. 1.

```
Procedure GbSA
 $SG \leftarrow GenerateInitialSolution$
 $SG \leftarrow LocalSearch(SG)$
While (termination condition is not met) do
    $Flag \leftarrow False$
    $SpiralChaoticMove(SG, Flag)$
    If ( $Flag$ ) then
        $SG \leftarrow LocalSearch(SG)$
    Endif
Endwhile
Return $SG$
Endprocedure
```

Fig. 1. The pseudo-code of the proposed GbSA.

At first, the initial solution is created by function $GenerateInitialSolution$ of Fig. 1. For the proposed GbSA-MLT, with the assumption that the minimum gray-level can be zero and maximum gray level can be 255, the initial solution $S = \{S_1, S_2, ..., S_L\}$ is calculated by the following linear formula:

$$S_i = 1 + \left\lfloor i \cdot \frac{253}{L} \right\rfloor, i = \{1, 2, ..., L\} \quad (2)$$

Where L is the number of thresholds, and S_i denotes the value of threshold i .

Following solution (thresholds) initialization, the local search component of the GbSA, $LocalSearch(SG)$, is activated with the initial thresholds (solution) in variable SG . Here, the local search is a modified Hill-Climbing search algorithm whose pseudo-code is shown in Fig. 2.

The $LocalSearch$ in Fig. 2 is given a solution S with L components. Then, it seeks the nearest best solution to S by gradually increasing search step sizes with the constant parameter $\alpha\Delta S$. The value of α is also increased with the constant parameter $\Delta\alpha$ and a chaotic sequence. To sum up, $LocalSearch$ searches the space around the given solution S with small step sizes. Then, it gradually increases the step sizes to faster explore the search space. At the end, it returns the locally-best solution found around the given solution S as shown in Fig. 1.

Other components of the proposed GbSA are called in the "”while" loop of the pseudo-code in Fig. 1. $SpiralChaoticMove$ is the first component in the loop which globally searches around the solution SG . It stops searching whenever it reaches a solution better than SG or it exceeds the maximum repetition number denoted by the user-selected parameter $Max\,\mathrm{Re}\,p$. If $SpiralChaoticMove$ finds a better solution, $Flag$ is set to *true* and $LocalSearch$ is called to search locally around the newly-updated solution SG . The whole process above is repeated until a stopping condition is satisfied. Here, the "while" loop stops when no further chance is observed in the Otsu's criterion.

The pseudo-code of $SpiralChaoticMove$ is shown in Fig. 3. The current best solution, denoted by S in Fig. 3, is given to $SpiralChaoticMove$. Then, each component S_i of S is modified by

$$SNext_i \leftarrow S_i \pm NextChaos() \cdot r \cdot \cos(\theta_i) \quad (3)$$

$SNext_i$ is the ith component of the next solution $SNext$, which is on the arm of the spiral galaxy having core S .

The function $NextChaos()$ returns a chaotic number between zero and one, which is generated by the logistic map. The logistic map is a one-dimensional noninvertible map which is able to generate chaotic sequences:

$$x_{n+1} = \lambda x_n(1 - x_n), n = 0,1,2,... \quad (4)$$

The initial value x_0 should be chosen from $[0,1]$. λ is the control parameter, and x_n denotes the variable at discrete time n. The logistic map exhibits chaotic dynamics when $\lambda = 4$ and $x_0 \in [0,1] - \{0, 0.25, 0.5, 0.75, 1\}$.

In Figs. 2 and 3, the symbol $f(.)$ denotes the Otsu's criterion, which the GbSA tries to maximize. It is mentioned that the proposed GbSA stops when the two successive values of $f(.)$ are the same.

```
Procedure LocalSearch
// input:
    L is the number of components of candidate
solutions.
    S is the current solution with L components such
    that S_i denotes the component ith of solution S.
// output:
    SNext is the output of the local search
// parameters:
    ΔS is the step size which is set by
    function NextChaos().
    α is a dynamic parameter .
    Δα is 0.5.
    KMax denotes the maximum iteration that the local
    search has to search around a component to find a
    better solution. Here, 100.
Repeat for $i = 1 \; to$ L
    $\alpha = 1; k \leftarrow 0$
    While $k < kMax$
        $SL_i \leftarrow S_i - \alpha \cdot \Delta S$ ; $SU_i \leftarrow S_i + \alpha \cdot \Delta S$
        If $f(SL) < f(S)$ and $f(SU) < f(S)$ then
            Goto Endrepeat
        Endif
        If $f(SU) > f(S)$ then
            $S_i \leftarrow SU_i; SL_i \leftarrow SU_i; \alpha \leftarrow 1; k \leftarrow 0$
        Endif
        If $f(SL) > f(S)$ then
            $S_i \leftarrow SL_i; SU_i \leftarrow SL_i; \alpha \leftarrow 1; k \leftarrow 0$
        Endif
        $\alpha \leftarrow \alpha + \Delta\alpha \cdot NextChaos(); k \leftarrow k + 1$
    Endwhile
    $SL_i \leftarrow S_i; SR_i \leftarrow S_i$
Endrepeat
$SNext \leftarrow S$
Endprocedure
```

Fig. 2. The pseudo-code of the local search used in the GbSA.

```
Procedure SpiralChaoticMove
// input:
        L is the number of components of solutions.
        S is the current best solution with L components
        such that S_i denotes the ith component of
        solution S.
// output:
        SNext is the output, which is found first that is
        better than the given solution S.
        Flag is set to true to indicate that a better
        solution has been found.
// parameters:
        Δθ is a parameter. Here, 0.01.
        r is 0.001.
        Δr is set by function NextChaos() in each call.
        MaxRep is the maximum iteration that the
        SpiralChaoticMove searches. Here, 500.

Repeat $i = 1 \; to \; L$
$\theta_i \leftarrow (-1 + 2 \cdot NextChaos\;()) \cdot \pi$ ;
Endrepeat
While $rep < Max\,\mathrm{Re}\,p$
    Repeat for $i = 1 \; to \; L$
        $SNext_i \leftarrow S_i + NextChaos() \cdot r \cdot \cos(\theta_i)$ ;
    Endrepeat
    If ( $f(SNext) \geq f(S)$ ) then
        $Flag \leftarrow true$; Goto Endprocedure;
    Endif
    Repeat for $i = 1 \; to \; L$
        $SNext_i \leftarrow S_i - NextChaos() \cdot r \cdot \cos(\theta_i)$ ;
    Endrepeat
    If ( $f(SNext) \geq f(S)$ ) then
        $Flag \leftarrow true$; Goto Endprocedure;
    Endif
    $r \leftarrow r + \Delta r$ ;
    Repeat $i = 1 \; to \; L$
        $\theta_i \leftarrow \theta_i + \Delta\theta$ ;
        If( $\theta_i > \pi$ ) then   $\theta_i \leftarrow -\pi$ ;
        Endif
    Endrepeat
    $rep \leftarrow rep + 1$ ;
Endwhile
Endprocedure
```

Fig. 3. The pseudo-code of the SpiralChaoticMove used in the GbSA.

IV. EXPERIMENTAL RESULTS

In this section, several well-known images [9] are employed for testing the proposed GbSA-MLT. Three images are used for the experiments: Lena, Peppers, and Baboon; as shown in Fig. 4. They are originally color images and they are converted to gray images by replacing each color pixel (R, G, B) with its intensity, which is obtained by $0.299R + 0.587G + 0.114B$.

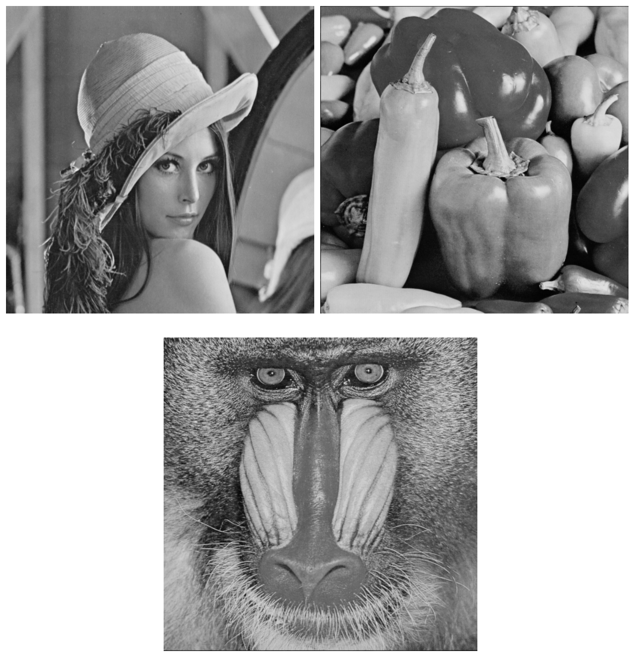
Fig. 4. The test images Lena, Peppers, and Baboon.

The gray-level test images are thresholded by one, two, three, four, five, and six thresholds obtained by the proposed GbSA-MLT, which are shown in Figs. 5 to 7.

The thresholds are compared with those obtained by the exhaustive search for optimizing Otsu's criterion, which are shown in Table 1. It is seen that for seven out of nine cases, the thresholds are exactly the same. Moreover, for the two cases, the thresholds are very close. The exhaustive search finds the optimum value 2683.66 and 1638.18 for Peppers and Baboon with three thresholds, respectively; whereas the GbSA-MLT converges to 2683.56 and 1638.07, respectively. In other words, the GbSA_MLT finds optimal or very near-optimal solutions for the Otsu's criterion. It is worth mentioning that the GbSA_MLT always finds the same thresholds for the images in the runs that have been tested (10 runs for each image).

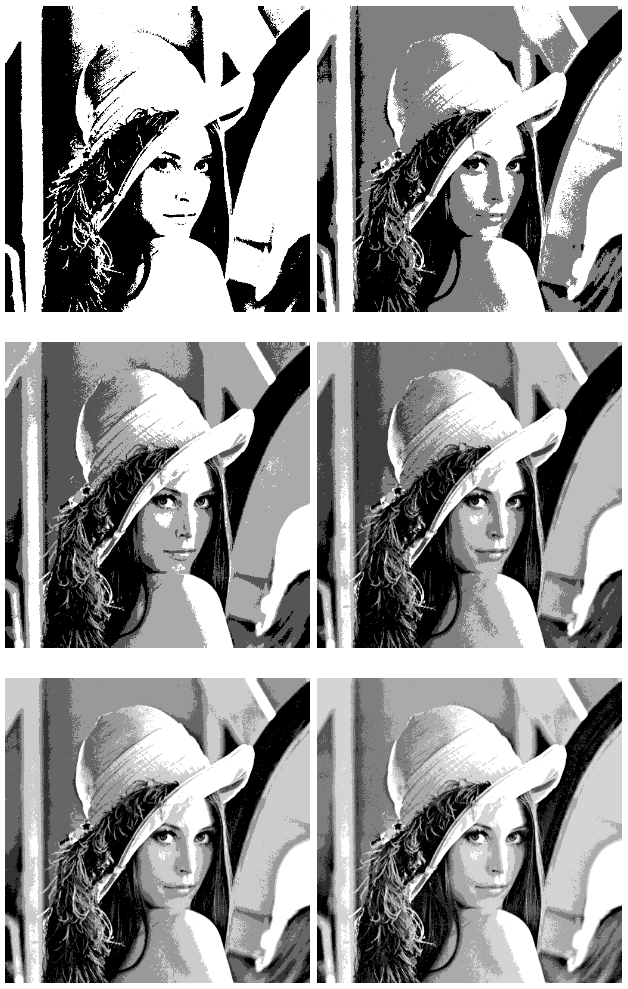
Fig. 5. Lena image thresholded by one to six thresholds from top left to bottom right, respectively using the proposed GbSA.

The time comparison between the GbSA-MLT and the exhaustive search for Otsu's criterion is expressed in Table 2. Only for one threshold, the exhaustive search is slightly better than the GbSA_MLT and as the number of thresholds increases, the time to find the optimal thresholds grows rapidly such that for cases above three thresholds the computation time is beyond hours and even days. In contrast, according to Table 2, the GbSA_MLT always converges in average of ten runs below 0.34 seconds. It is noted that the experiments are performed on a PC with a Pentium IV CPU running Microsoft Windows XP Operating System.

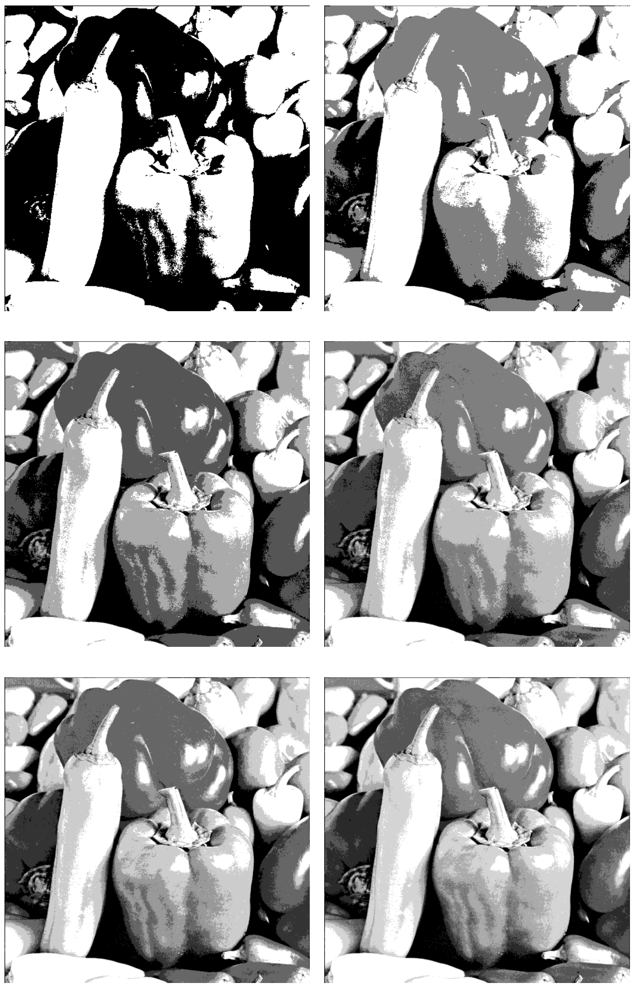

Fig. 6. Peppers image thresholded by one to six thresholds from top left to bottom right, respectively using the proposed GbSA.

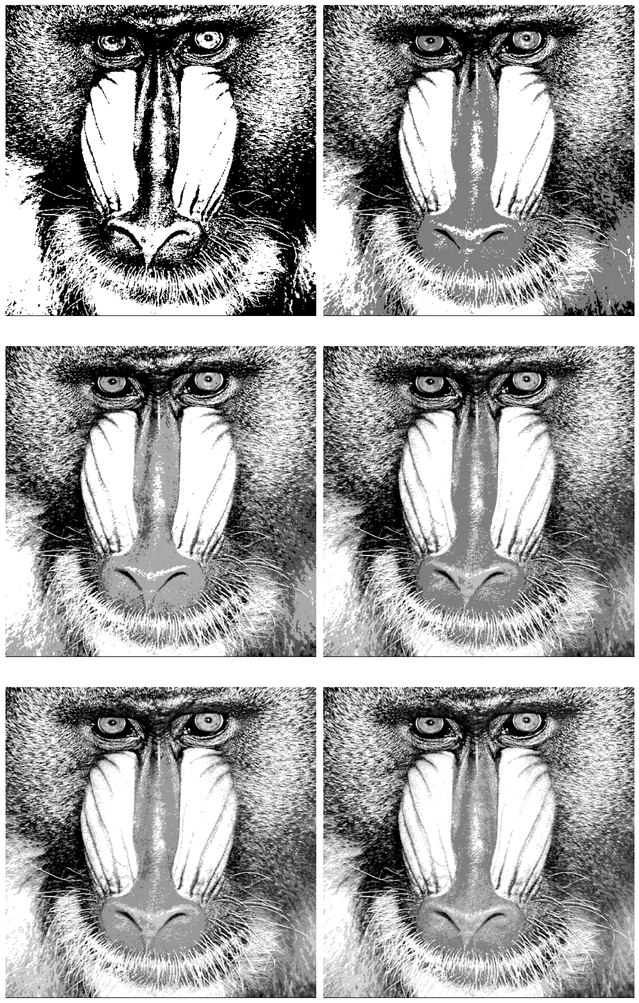

Fig. 7. Baboon image thresholded by one to six thresholds from top left to bottom right, respectively using the proposed GbSA.

V. COCLUSIONS

Here, a newly introduced metaheuristic called "Galaxy-based Search Algorithm" or GbSA is used for multilevel thresholding. The GbSA imitates the arms of spiral galaxies to look for optimal solutions. The GbSA also utilizes a local search algorithm for fine-tuning of the solutions obtained by the spiral arm of the GbSA. In addition, chaotic maps are employed inside the GbSA's components.

The GbSA is proposed here to optimize the Otsu's criterion for multilevel thresholding of gray-level images. The performance of the GbSA for multilevel thresholding is compared with an exhaustive search based on the Otsu's criterion. The experiments demonstrate that the proposed GbSA is so promising in this application. However, more experiments need to be done in this regard and the GbSA should be used for other applications.

Table 1. The proposed GbSA-MLT thresholds are compared with those obtained by the exhaustive search. For one and two thresholds, the solutions are the same.

IMAGE	GSA-MLT / OTSU THRESHOLDS		
	one	two	three
Lena	116	90, 148	77,122,166
Peppers	118	66,133	60,114,161 / 61,115,162
Baboon	126	94, 146	76,114, 153 / 78, 116, 154

Table 2. The proposed GbSA-MLT time (in seconds) versus exhaustive search for Lena image, using one to six thresholds.

GSA-MLT / OTSU TIME					
One	***two***	***three***	***four***	***five***	***Six***
0.07/0.02	0.12/0.7	0.17/60.2	0.26/---	0.31/---	0.34/---

REFERENCES

[1] N. Otsu N, "A threshold selection method from gray-level histogram," IEEE Trans. Systems Man Cybern. Vol. 9, no. 1, 1979, pp. 62–66.

[2] H. Gao H, W. Xu, J. Sun, and Y. Tang, "Multilevel thresholding for image segmentation through an improved quantum-behaved particle swarm algorithm," IEEE Transactions On Instrumentation And Measurement, vol. 59, no. 4, 2010, pp. 934-946.

[3] H. Shah-Hosseini, "Problem solving by intelligent water drops," Proceedings of IEEE Congress on Evolutionary Computation, Swissotel The Stamford, Singapore, 2007, pp. 3226-3231.

[4] H. Shah-Hosseini, "Intelligent water drops algorithm: a new optimization method for solving the multiple knapsack problem," Int. Journal of Intelligent Computing and Cybernetics, 1(2), 2008, pp. 193-212.

[5] H. Shah-Hosseini, "The Intelligent Water Drops algorithm: A nature-inspired swarm-based optimization algorithm," Int. J. Bio-Inspired Computation, 1(1/2), 2009, pp. 71–79.

[6] H. Shah-Hosseini, "Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimisation," Int. J. Computational Science and Engineering, 6(1/2), pp. 132-140.

[7] E.H.L. Aarts and J.K. Lenstra, "Local search in combinatorial optimization," in Discrete Mathematics and Optimization, Eds. Wiley, Chichester, UK, 1997.

[8] N. Mladenovic and P. Hansen, "Variable neighborhood search," Computers Operational Research. vol. 24, 1997, pp. 1097-1100.

[9] USC-SIPI Image Database,.Available at: http://sipi.usc.edu/database, 2011.