Proceedings of the International Multiconference on Computer Science and Information Technology pp. 735–740
ISBN 978-83-60810-27-9
ISSN 1896-7094

A Graphical Interface for Evaluating Three Graph-Based Image Segmentation Algorithms

Gabriel Mihai
University of Craiova, Faculty of Automation, Computers and Electronics, Craiova, Romania
Email:mihai_gabriel@software.ucv.ro

Alina Doringa, Liana Stanescu
University of Craiova, Faculty of Automation, Computers and Electronics, Craiova,Romania
Email:alinadoringa@hotmail.com, stanescu_liana@software.ucv.ro

Abstract—**Image segmentation has an essential role in image analysis, pattern recognition and low-level vision. Since multiple segmentation algorithms exists in literature, numerical evaluations are needed to quantify the consistency between them. Error measures can be used for consistency quantification because are allowing a principled comparison between segmentation results on different images, with differing numbers of regions, and generated by different algorithms with different parameters. This paper presents a graphical interface for evaluating three graph-based image segmentation algorithms: the color set back-projection algorithm, an efficient graph based image segmentation algorithm known also as the local variation algorithm and a new and original segmentation algorithm using a hexagonal structure defined on the set of image pixels.**

I. Introduction

IMAGE segmentation is one of the most difficult and challenging tasks in image processing and can be defined as the process of dividing an image into different regions such that each region is homogeneous while not the union of any two adjacent regions.

The consistency between segmentations must be evaluated because no unique segmentation of an image can exist. If two different segmentations arise from different perceptual organizations of the scene, then it is fair to declare the segmentations inconsistent [3].

This paper presents a graphical interface used for an objective and quantitative evaluation of three graph-based segmentation algorithms that will be described further. For each of these algorithms three characteristics are examined [4]: correctness, stability with respect to parameter choice, stability with respect to image choice.

The evaluation of the algorithms is based on two metrics (GCE, LCE) defined in [3] which can be used to measure the consistency of a pair of segmentations. These measures allows a comparison between segmentation results on different images, with differing numbers of regions, and generated by different algorithms with different parameters. In order to establish which algorithm produces better results these are compared with manual segmentations of the same image.

For searching certain structures graph-based segmentation methods such as minimum spanning tree [6] [7], or minimum cut [8] [9] are using an edge weighted graph constructed on the image pixels. The Graph-based clustering algorithms use the concept of homogeneity of regions. For color segmentation algorithms the homogeneity of regions is color-based, and thus the edge weights are based on color distance.

For obtaining the image regions early graph-based methods have used fixed thresholds and local measures. Using these values larger edges belonging to a minimum spanning tree were beaked. The problem of fixed threshold [10] can be avoided by determining the normalized weight of an edge using the smallest weight incident on the vertices touching that edge.

Other methods presented in [6] [7] use an adaptive criterion that depends on local properties rather than global ones. The methods based on minimum cuts in a graph are designed to minimize the similarity between pixels that are being split [8], [9]. An alternative to the graph cut approach is to look for cycles in a graph embedded in the image plane [11].

The color and texture models are used by most graph-based segmentation approaches but these homogeneity criteria have some drawbacks for object extraction.

In [12] a source of additional information denoted by the term of syntactic features is presented, which represent geometric properties of regions and their spatial configurations such as homogeneity, compactness, regularity, inclusion or symmetry.

II. The color set back-projection algorithm

The color set back-projection algorithm proposed in [5] is a technique for the automated extraction of regions and representation of their color content. This algorithm is based on color sets which provide an alternative to color histograms for representing color information. Their utilization is possible only when salient regions have few equally prominent colors. Each pixel from the initial image is represented in the HSV color space. The quantized colors from 0 to 165 are stored in a matrix that is filtered by a 5x5 median filter for eliminating the isolated points. The back-projection process requires several stages:

a) Color set selection - candidate color sets are selected first with one color, then with two colors, etc., until the salient regions are extracted

b) Back-projection onto the image - a transformation from the RGB color space to HSV color space and a quantization of the HSV color space at 166 colors is performed for each segmented image.

c) Thresholding and labeling image - insignificant color information is reduced and the significant color regions are evidentiated, followed by the generation, in automatic way, of the regions of a single color, of the two colors, of three colors.

In the implementation of the color set back-projection algorithm that can be found it in [1] [13]. After processing the global histogram of the image, and the color set are provided. The process of regions extraction is using the filtered matrix and it is a depth – first traversal described in pseudo-cod in the following way:

```
Procedure FindRegions (Image I, colorset C)
1) InitStack(S)
2) Visited = ∅
3) for *each node P in the I do
4) if *color of P is in C then
5)    PUSH(P)
6)    Visited ← Visited ∪ {P}
7)    while not Empty(S) do
8)      CrtPoint ←POP()
9)      Visited ← Visited ∪ {CrtPoint}
10)      For *each unvisited neighbor S of CrtPoint do
11)       if *color of S is in C then
12)          Visited ← Visited ∪ {S}
13)          PUSH(S)
14)      end
15)    end
16)  end
17) *Output detected region
18)  end
19) end
```

The total running time for a call of the procedure FindRegions (Image *I*, colorset *C*) is $O(m^2 * n^2)$ where *m* is the width and *n* is the height of the image [1][13].

III. Local Variation Algorithm

This algorithm described in [6] is using a graph based approach for the image segmentation process. The pixels are considered the graph nodes so in this way it is possible to define an undirected graph G = (V, E) where the vertices v_i from V represent the set of elements to be segmented. Each edge (v_i, v_j) belonging to E has associated a corresponding weight $w(v_i, v_j)$ calculated based on color, which is a measure of the dissimilarity between neighboring elements v_i and v_j.

A minimum spanning tree is obtained using Kruskal's algorithm. The connected components that are obtained represent image's regions. It is supposed that the graph has *m* edges and *n* vertices. This algorithm is described also in [14] where it has four major steps that are presented below:

1) Sort $E = (e_1, ..., e_m)$ such that $|e_t| < |e_{t'}| \ \forall t < t'$
2) Let $S^o = (\{x_1\}, ..., \{x_n\})$ be each initial cluster containing only one vertex.
3) For $t = 1, ..., m$
   a) *Let* x_i *and* x_j *be the vertices connected by* e_t
   b) *Let* $C_{x_i}^{t-1}$ *be the connected component containing point* x_i *on iteration t-1 and* $l_i = max_{mst} C_{x_i}^{t-1}$ be the longest edge in the minimum spanning tree of $C_{x_i}^{t-1}$ .Likewise for l_j.
   c) *Merge* $C_{x_i}^{t-1}$ *and* C_{xj}^{t-1} *if* $|e_t| < \min\{ l_i + \frac{k}{C_{x_i}^{t-1}}, l_j + \frac{k}{C_{x_j}^{t-1}} \}$ *where k is a constant.*
4) $S = S^m$

The existence of a boundary between two components in segmentation is based on a predicate D. This predicate is measuring the dissimilarity between elements along the boundary of the two components relative to a measure of the dissimilarity among neighboring elements within each of the two components. The internal difference of a component C ⊆ V was defined as the largest weight in the minimum spanning tree of a component MST(C, E):

$Int(C) = \max_{e \in MST(C,E)} w(e)$. The difference between two components $C_1, C_2 \subseteq V$ is defined as the minimum weight edge connecting the two components:

$$Dif(C_1, C_2) = \min(w(v_i, v_j)) \quad v_i \in C_1, v_j \in C_2, (v_i, v_j) \in E$$

A threshold function is used to control the degree to which the difference between components must be larger than minimum internal difference. The pair wise comparison predicate is defined as:

$$D(C_1, C_2) = \begin{cases} true \text{ if } Dif(C_1, C_2) > MInt(C_1, C_2) \\ false \text{ otherwise} \end{cases}$$

where the minimum internal difference Mint is defined as:

$MInt(C_1, C_2) = \min(Int(C_1) + \tau(C_1), Int(C_2) + \tau(C_2))$.

The threshold function was defined based on the size of the component: $\tau(C) = k / |C|$.The k value is set taking into account the size of the image. The algorithm for creating the minimum spanning tree can be implemented to run in $O(m \log m)$ where m is the number of edges in the graph.

IV. Image Segmentation Using a Hexagonal Structure Defined on the Set of Pixels

The technique is based on a new and original utilization of pixels from the image that are integrated into a network type graph [2]. The hexagonal network structure on the image pixels, as presented in figure 1 was selected to improve the

running time required by the algorithms used for segmentation and contour detection.

The hexagonal structure represents a grid-graph and for each hexagon h in this structure there exist 6-hexagons that are neighbors in a 6-connected sense. The time complexity of the algorithms is reduced by using hexagons instead of pixels as elementary piece of information. The index of each hexagon is stored in a vector of numbers $[1..N]$, where N, the number of hexagons, is calculated using the formula [2]:

$$N = \frac{height - 1}{2} * (\frac{width - width \bmod 4}{2} - 1)$$

where *height* and *width* represent the height and the width of the image.

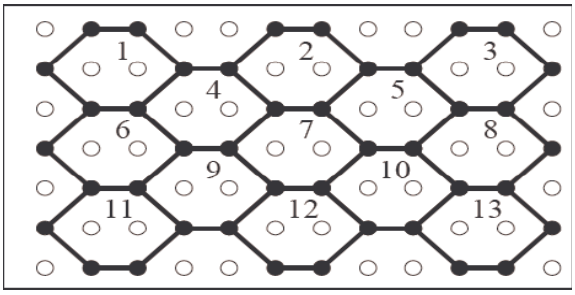

Fig.1. Hexagonal structure constructed on the image pixels

With each hexagon two important attributes are associated:

a) The dominant color - eight pixels contained in a hexagon are used: six pixels from the frontier and two from interior

b) The gravity center

The pixels of image are split into two sets, the set of pixels representing the vertices of hexagons and the set of complementary pixels. These two lists are used as inputs for the segmentation algorithm.

Based on the algorithms proposed in [2] the list of salient regions is obtained from the input image using the hexagonal network and the distance between two colors in HSV space color. The color of a hexagon is obtained using a procedure called *sameVertexColour*. The execution time of this procedure is constant because all calls are constant in time processing. The color information is used by the procedure *expandColorArea* to find the list of hexagons that have the same color. To expand the current region it is used a procedure containing as input:

a) The current hexagon h_i ,

b) L_1 and L_2 lists,

c) The list of hexagons V

d) The current region index *indexCrtRegion*

e) The current color index *indexCrtColor*.

The output is represented by a list of hexagons with the same color *crtRegionItem*. The running time of the procedure *expandColourArea* is $O(n)$ where n is the number of hexagons from a region with the same color.

The list of regions is obtained using the *listRegions* procedure The input of this procedure contains:

a) The vector V representing the list of hexagons

b) The lists L_1 and L_2

The output is represented by a list of colors pixels and a list of regions for each color.

Procedure **listRegions** (V, L_1, L_2)

1) colourNb ←0;
2) for i ←1 to n do
3) initialize crtRegionItem;
4) if not(visit(h_i)) then
5) crtColorHexagon ←sameVertexColour (L_1, L_2, h_i);
6) if crtColorHexagon.sameColor then
7) k ← findColor(crtColorHexagon.color);
8) if k < 0 then
9) add new color *ccolourNb* to list C;
10) k ←colourNb++;
11) indexCrtRegion ←0;
12) else
13) indexCrtColor ←k;
14) indexCrtRegion ←findLastIndexRegion(index CrtColor);
15) indexCrtRegion++;
16) End
17) hi.indexRegion ←indexCrtRegion;
18) hi.indexColor ←k;
19) add h_i to crtRegionItem;
20) expandColourArea(h_1, L_1, L_2, V ,indexCrtRegion, indexCrtColor; crtRegionItem);
21) add new region crtRegionItem to list of element k from C
22) end
23) end
24) end

The running time of the procedure list Regions is $O(n^2)$, where n is the number of the hexagons network [2].

V. SEGMENTATION ERROR MEASURES

To evaluate a segmentation algorithm it is needed to measure the accuracy, the precision and the performance. When multiple segmentation algorithms are evaluated some metrics are needed to establish which algorithm produce better results.

In [3] are proposed two metrics tolerant to refinement that can be used to evaluate the consistency of a pair of segmentations. A segmentation error measure takes two segmentations S_1 and S_2 as input, and produces a real valued output in the range $[0..1]$ where zero signifies no error. For a given pixel p_i two segments S_1 and S_2 containing that pixel, are considered. If one segment is a proper subset of the other, then the pixel lies in an area of refinement, and the local error should be zero. If there is no subset relationship, then the two regions overlap in an inconsistent

manner. In this case, the local error should be non-zero. Let \ denote set difference, and $|x|$ the cardinality of set x. If $R(S,p_i)$ is the set of pixels corresponding to the region in segmentation S that contains pixel p_i, the local refinement error is defined in [3] as:

$$E(S_1,S_2,p_i)=\frac{|R(S_1,p_i)\setminus R(S_2,p_i)|}{R(S_1,p_i)}$$

This local error measure is not symmetric and it encodes a measure of refinement in one direction only. Given this local refinement error in each direction at each pixel, there are two natural ways to combine the values into an error measure for the entire image. Global Consistency Error (GCE) forces all local refinements to be in the same direction. Local Consistency Error (LCE) allows refinement in different directions in different parts of the image. Let n be the number of pixels [3]:

$$GCE(S_1,S_2)=\frac{1}{n}\min\{\ {}_iE(S_1,S_2,p_i),E(S_2,S_1,p_i)\}$$

$$LCE(S_1,S_2)=\frac{1}{n}\ {}_i\min(E(S_1,S_2,p_i),E(S_2,S_1,p_i))$$

$LCE \le GCE$ for any two segmentations and it is clear that GCE is a tougher measure than LCE. In [3] are shown that, as expected, when pairs of human segmentations of the same image are compared, both the GCE and the LCE are low; conversely, when random pairs of human segmentations are compared, the resulting GCE and LCE are high. If the pixel wise minimum is replaced by a maximum it is obtained a new measure named Bidirectional Consistency Error (BCE) that is not tolerating the refinement. This measure is evaluated using

$$BCE(S_1,S_2)=\frac{1}{n}\ {}_i\max(E(S_1,S_2,p_i),E(S_2,S_1,p_i)).$$

If an image is interpreted as a set O of pixels and the segmentation as a clustering of O we can apply measures for comparing clusters. As described in [17] we can have two types of distances available for clusters: the distance between clusters evaluated by counting pairs and the distance between clusters evaluated using set matching. For the first case are assumed two clusters C_1 and C_2 belonging to a set O of objects and also all pairs of distinct objects (o_i,o_j) from OxO.

Each pair can be found into one of the four categories:

a) in the same cluster under both C_1 and C_2 (the total number of these pairs is represented by N_{11})

b) in different clusters under both C_1 and C_2 (N_{00}),

c) in the same cluster under C_1 but not C_2 (N_{10}),

d) in the same cluster under C_2 but not C_1 (N_{01}).

Based on these assumptions the following statement is obtained: $N_{11}+N_{00}+N_{10}+N_{01}=n(n-1)/2$.

Based on these numbers multiple distances of the first type were defined. One of the distances described in [18] is the Rand index evaluated as $R(C_1,C_2)=1-\frac{N_{11}+N_{00}}{n(n-1)/2}$.

A perfect matching between two clusters is implied by a 0 value. Another index called the Jacard index [19] is evaluated as $J(C_1,C_2)=1-\frac{N_{11}}{N_{11}+N_{10}+N_{01}}$

For the second type the comparison criteria is based on the following term $a(C_1,C_2)=\ {}_{ci\in C_1}\max_{cj\in C_2}|c_i\cap c_j|$.

This term measures the matching degree between C_1 and C_2 has a value of n when the two clusters are equal. The Dongan index [20] is evaluated as $D(C_1,C_2)=2n-a(C_1,C_2)-a(C_2,C_1)$.

VI. Experimental results

For each image three major steps are required in order to calculate GCE, LCE, BCE, Dongen index and Rand index values:

a) Obtain the image regions by applying segmentation algorithms (color set back-projection – CS, the algorithm based on the hexagonal structure – HS, the local variation algorithm – LV) and the regions manually segmented -MS

b) Store the information obtained for each region in the database

c) Use a graphical interface to easily calculate GCE and LCE values and to store them in the database for later statistics

All algorithms are using a configurable threshold value to keep only the regions that have a number of pixels greater than this value. In this way only representative regions are saved in the database and used later in the experiments.

The experiments were made on a working set containing 300 images selected from IAPR-TC12 [15] and Berkeley [16] segmentation datasets. All values were calculated using the graphical interface presented in figure 1.

This application is offering to the user a list containing the name of all images that were segmented and that have regions stored in the database. After selecting from the list the name of an image the user needs to press *GetInfo* button to retrieve de information stored in the database for that image. In that moment the text boxes containing the number of regions obtained with each algorithm are filled.

For each algorithm a button named *ViewRegions* is available. By pressing this button a new form is shown containing the obtained regions as presented in image 2. This form helps the user to view the regions detected by each algorithm and to make an empirical evaluation of the performance. Because this evaluation is subjective we need a numerical evaluation.

Since the information about regions is available the user can press Calculate button. In this moment the error

measures will be calculated and shown to the user. If the user decides that these values are needed later he can choose to store them in the database by pressing *Store results* button. To compare the obtained results with previous results stored in the database *View stored results* button should be pressed. In this way it is possible to evaluate what algorithm is producing better results.

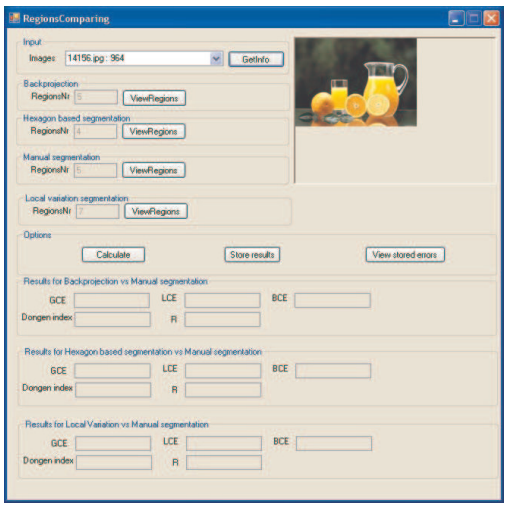

Fig.1. The graphical interface used for errors calculation

Fig.2. The form containg image regions

In table 1 are presented the images for which we will present some experimental results.

TABLE I. IMAGES USED IN EXPERIMENTS

Images		
1 	2 	3 
4 	5 	6 

In table 2 can be seen the number of regions resulted from the application of the segmentation algorithms.

TABLE II .THE NUMBER OF REGIONS DETECTED FOR EACH ALGORITHM

Img.No	CS	HS	LV	MS
1	4	3	6	5
2	5	4	7	6
3	5	3	5	3
4	6	4	6	3
5	4	2	5	2
6	5	2	6	3

In table 3 are presented the GCE values calculated for each algorithm.

TABLE III .GCE VALUES CALCULATED FOR EACH ALGORITHM

Img.No	GCE - CS	GCE-HS	GCE - LV
1	0.18	0.09	0.24
2	0.36	0.10	0.28
3	0.18	0.09	0.11
4	0.10	0.09	0.09
5	0.11	0.09	0.10
6	0.04	0.02	0.03

In table 4 are presented the LCE values calculated for each algorithm.

TABLE IV. LCE VALUES CALCULATED FOR EAC H ALGORITHM

Img.No.	LCE-CS	LCE-HS	LCE -LV
1	0.11	0.07	0.15
2	0.18	0.12	0.17
3	0.17	0.07	0.08
4	0.09	0.11	0.09
5	0.05	0.08	0.10
6	0.04	0.02	0.02

Bellow is presented the regions obtained for the third image after applying manual segmentation and the three algorithms presented above.

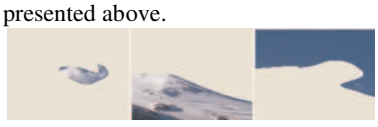

Fig.3. Regions from manual segmentation

Fig.4.Regions from the color set back-projection algorithm

Fig.5. Regions from hexagonal structure segmentation

Fig.6. Regions from the local variation algorithm

The error measures presented in tables three and four are calculated in relation with manual segmentation which is considered the ground truth. It can be observed that the values for GCE and LCE are lower in case of hexagonal segmentation. For example for the first analyzed image GCE = 0.09 and LCE =0.07 in the case of hexagonal segmentation, GCE=0.18 and LCE=0.11 for the color set back-projection algorithm, GCE=0.24 and LCE=0.15 for the local variation algorithm. The error measures, for almost all tested images, have smaller values in case of the hexagonal segmentation so this algorithm is a good refinement of the manual segmentation.

VI. Conclusion

In this paper it is proposed a graphical interface for evaluating three graph-based image segmentation algorithms: the color set back-projection algorithm, the image segmentation using a hexagonal structure defined on the set of image pixels and the local variation algorithm. Error measures like GCE, LCE are used to evaluate the accuracy of each segmentation produced by the algorithms. The values for the error measures are calculated using the described application specially created for this purpose. The proposed error measures quantify the consistency between segmentations of differing granularities. Because human segmentation is considered truth segmentations the error measures are calculated in relation with manual segmentation. The GCE and LCE demonstrate that the image segmentation based on a hexagonal structure produces a better segmentation than the other methods. In the future work the comparative study will be effectuated on medical images.

IV. References

[1] D. D. Burdescu, L. Stanescu "A New Algorithm for Content-Based Region Query in Multimedia Databases" Congres DEXA 2005 : Database and expert systems applications Copenhagen, 22-26 August 2005 , vol. 3588, pp. 124-133.

[2] D. D. Burdescu, M. Brezovan, E. Ganea, and L. Stanescu "A New Method for Segmentation of Images Represented in a HSV Color Space" ACIVS 2009: Bordeaux, France, pp. 606-617 .

[3] D. Martin, C. Fowlkes, D. Tal, J. Malik "A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics " In IEEE (ed.), Proceedings of the Eighth International Conference On Computer Vision (ICCV-01), July 7-14, 2001, Vancouver, British Columbia, Canada, vol. 2, pp. 416–425.

[4] R. Unnikrishnan, C. Pantofaru, and M. Hebert. "Toward Objective Evaluation of Image Segmentation Algorithms", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 29, No. 6, June, 2007, pp. 929-944.

[5] J. R. Smith, S. F. Chang.: "Tools and Techniques for Color Image Retrieval", Symposium on Electronic Imaging. In: Science and Technology - Storage & Retrieval for Image and Video Databases IV, volume 2670, San Jose, CA, February 1996. IS&T/SPIE. (1996)

[6] P.F. Felzenszwalb, W.D. Huttenlocher: "Efficient Graph-Based Image Segmentation", Intl. Journal of Computer Vision, 59(2), pp. 167–181 (2004)

[7] L. Guigues, , L.M Herve, L.-P. Cocquerez: "The hierarchy of the cocoons of a graph and its application to image segmentation." Pattern Recognition Letters, 24(8), pp. 1059–1066 (2003)

[8] J. Shi, J. Malik: "Normalized cuts and image segmentation", Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, San Juan, Puerto Rico, pp. 731–737 (1997)

[9] Z. Wu ,R. Leahy: "An optimal graph theoretic approach to data clustering: theory and its application to image segmentation", IEEE Trans. on Pattern Analysis and Machine Intelligence, 15(11),pp. 1101–1113 (1993)

[10] R. Urquhar: "Graph theoretical clustering based on limited neighborhood sets." Pattern Recognition, 15(3), pp 173–187 (1982)

[11] I. Jermyn, H. Ishikawa: "Globally optimal regions and boundaries as minimum ratio weight cycles." IEEE Trans. on Pattern Analysis and Machine Intelligence, 23(8), pp. 1075–1088 (2001)

[12] C.F. Bennstrom, J.R. Casas: "Binary-partition-tree creation using a quasi-inclusion criterion." Proc. of the Eighth International Conference on Information Visualization, London, UK, pp. 259–294, (2004)

[13] L. Stanescu "Visual information. Procecessing, Retrieval and Applications" Sitech Craiova 2008

[14] C.Pantofaru, M.Hebert: "A Comparison of Image Segmentation Algorithms". Technical Report CMU-RI-TR-05-40, Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania .(2005).

[15] ImageClef (2009) http://imageclef.org/2010

[16] http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/BSDS300/html/dataset/images.html

[17] X Jiang, C Marti, C Irniger, H Bunke (2005) Distance Measures for Image Segmentation Evaluation. In: EURASIP Journal on Applied Signal Processing, vol. 2006, pp. 1—10

[18] W. M. Rand "Objective criteria for the evaluation of clustering methods" In: Journal of the American Statistical Association, vol. 66, 1971, no. 336, pp. 846–-850

[19] A. Ben-Hur, A. Elisseeff, I. Guyon "A stability based method for discovering structure in clustered data" In: Proceedings of 7th Pacific Symposium on Biocomputing (PSB '02), vol. 7, 2002, pp. 6–-17, Lihue, Hawaii, USA

[20] S. van Dongen "Performance criteria for graph clustering and Markov cluster experiments" In: Tech. Rep. INS-R0012,10 EURASIP Journal on Applied Signal Processing Centrum voor Wiskunde en Informatica (CWI), 2000, Amsterdam