On Graph Extraction from Image Data

Andreas Holzinger, Bernd Malle, and Nicola Giuliani

Research Unit Human-Computer Interaction, Institute for Medical Informatics, Statistics & Documentation, Medical University Graz, Austria
{a.holzinger,b.malle,n.giuliani}@hci4all.at

Abstract. Hot topics in knowledge discovery and interactive data mining from natural images include the application of topological methods and machine learning algorithms. For any such approach one needs at first a relevant and robust digital content representation from the image data. However, traditional pixel-based image analysis techniques do not effectively extract, hence represent the content. A very promising approach is to extract graphs from images, which is not an easy task. In this paper we present a novel approach for knowledge discovery by extracting graph structures from natural image data. For this purpose, we created a framework built upon modern Web technologies, utilizing HTML canvas and pure Javascript inside a Web-browser, which is a very promising engineering approach. Following on a short description of some popular image classification and segmentation methodologies, we outline a specific data processing pipeline suitable for carrying out future scientific research. A demonstration of our implementation, compared to the results of a traditional watershed transformation performed in Matlab showed very promising results in both quality and runtime, despite some open problems. Finally, we provide a short discussion of a few open problems and outline some of our future research routes.

Keywords: data preprocessing, image segmentation, graphs, graph-based algorithms, graph extraction, image analysis, image content analytics, knowledge discovery, data mining.

1 Introduction and Motivation

Big challenges in the biomedical domain are today in the development of new methods, algorithms and tools for the effective analysis and interpretation of complex biomedical data [1]. Within such data sets, relevant structural and/or temporal patterns ("knowledge") are often hidden, difficult to extract, thus not directly accessible to a biomedical expert, consequently, a major challenge is in interactive Knowledge Discovery and Data Mining which relies heavily on machine learning approaches. However, many of the classical methods are based on the assumption that the data objects under consideration are represented in terms of feature vectors, or collections of attribute values; Bunke (2003) [2], for example, argued that graphs have a representational power that is significantly higher than the representational power of feature vectors. Moreover,

D. Ślęzak et al. (Eds.): BIH 2014, LNAI 8609, pp. 552–563, 2014.

graph-theory provides powerful tools to map data structures and to find novel connections between data objects [3] and allow the application of statistical and machine learning techniques [4].
Methods from computational geometry and algebraic topology may also be of great help [5], and could be combined with machine learning approaches, e.g. evolutionary algorithms [6], [7]. Promising future research routes in this field are in interactive visual data mining together with graph-based data analysis [8], [9]. Another benefit of a graph-based data structure is in the applicability of methods from network topology and network analysis and data mining, e.g. small-world phenomenon [10], and cluster analysis [11] to mention only two.

The application of graph theory to image analysis (see e.g. [12]) is in the focus of research for some time and still poses a lot of challenges and calls for new approaches.

2 Definitions

In this work we are dealing with *natural images*, which includes every digital image taken from real world scenes, for example biomedical images from dermoscopy (epiluminescence microscopy). The starting point of our calculations is the conversion of such a digital image into a topographic map, which we need for graph extraction. Caselles et al. (1999) [13] provide some necessary definitions:

Definition 1 (digital image). *A digital image is modelled as a real function $u(x)$, where x represents an arbitrary point of the plane and $u(x)$ denotes the grey-level at x. Let $u : \Omega \to \mathbb{R}$ be an image, i.e., a bounded measurable function.*

Definition 2 (upper level set). *Given an image u, we call upper level set of u any set of the form $[u \geq \lambda]$ where $\lambda \in \mathbb{R}$.*

Definition 3 (connected component). *Let X be a topological space. We say that X is connected if it cannot be written as the union ot two nonempty closed (open) disjoint sets. A subset C of X is called a connected component id C is a maximal connected subset of X, i.e., C is connected and for any conected subset C_1 of X such that $C \subseteq C_1$, then $C_1 = C$.*

Definition 4 (upper topographic map). *The upper topographic map of an image is the family of the connected components of the level sets od u, $[u \geq \lambda]$, $\lambda \in \mathbb{R}$.*

Definition 5 (topographic map). *If u belongs to a function space, such that each connected component of a level set is bounded by a countable or finite number of oriented Jordan curves, we call topographic map the family of these Jordan curves.*

3 Related Work

3.1 Traditional Image Classification

In the biomedical domain there has been a shift in demands, from software assisting in the production and processing of image data, analysed by humans alone, to software systems to represent, discover and evaluate knowledge. In [14] the authors describe a method of image classification which might be categorized as traditional, in the sense that it does not segment an image into logical substructures and attempts to compute relations among those. Instead, it uses metrics based on pixel values, divided into 1st order statistical parameters which describe the global structure of an image, and 2nd order statistical parameters which describe the neighborhoods of individual pixels.

- **1st order parameters.** Amongst the global parameters are those that use grey-value probabilities (histogram values) as the building blocks of their formulas. They include variance (as a measure of homogeneity), skewness (asymmetry of the value distribution), kurtosis (shape, either peaked or flat) and energy as well as entropy.
- **2nd order parameters.** In order to calculate those, the probability of grey-value co-occurrence is used as a basic concept. Using these values, the local / neighborhood parameters Energy, Entropy, Contrast, Homogeneity and Correlation are computed.

Once the needed metrics were taken, a C4.5 algorithm was used to build a decision tree and classify the images. Although the algorithm is not a segmentation approach as such and thus not immediately usable in the endeavour to extract graph structures out of image data, it could provide regional information usable as classifiers in tasks such as identifying structural / topographical primitives. Therefore it might constitute a building block in the preprocessing pipeline of a more extensive procedure.

3.2 Watershed Methods

Watershed algorithms [15] got their name from the fact that they treat images as topographic maps, that is as 'landscapes' with height structures. The segmentation of those landscapes into regions of pixels belonging together is then performed by assuming drops of water raining down on the map, following paths of descent into low areas until they form 'lakes', which in watershed terminology are called catchment basins. This can be thought of in one of various ways: by simulating drops raining from above, by immersing the whole landscape into an ocean (with holes punched into the deepest spots of the landscape, so the water can enter.), but also by using topographical distance measures like Minimum Spanning Trees (MST). A usual watershed processing pipeline consists of the following steps:

1. **Transformation into a topographic map.** The color or gray values of the pixels in an image are converted to height information; for example, if

given a grayscale from 0 (black) to 255 (white), one could assume 255 to be the highest possible peak in a landscape and 0 the deepest reachable point, thereby converting the image into a three-dimensional structure of voxels with coordinates (x, y, z). As a first step, the application of a gradient filter in order to produce continuous 'crests' throughout the landscape might be in order.

2. **Finding local minima.** In order to be able to fill a topographical relief with imaginary fluid by immersion, one first has to find the points through which the fluid can enter the landscape (or above which points it accumulates, depending on one's view). This is akin to finding the set of local minima in the image interpreted as topographical relief. This can easily be done in an image, by inspecting small regions of the image in sequence and finding the ones with the lowest value (one would probably use color or computed grey values). If such an operation would result in a significant percentage of all pixels marked as minima, which is often the case with watershed methods, a suitable subset of the computed minima (= seed points) can also be used instead.

3. **Finding catchment basins.** The main point in using watershed methods is finding regions that spatially belong together. They are usually seen as caverns or ditches separated by crests (the formations already extracted from color / grey value information earlier). This is done by using an algorithm to simulate flooding, so that the water accumulates in basins until it reaches a crest. At this point the basin can either 'flow over', filling an adjacent deep region as well, or the flooding can be stopped by erecting a virtual watershed. Another way to see this is by visualizing the voxels as vertices of a graph; based on this structure, the voxels belonging to a nearest minimum can be found by applying traditional, well-tested and well-understood graph algorithms such as Minimum Spanning Trees (MST). In order to find the voxels belonging to a catchment basin, the edges between them would first be assigned a weight corresponding to the distance they represent. By introducing auxiliary vertices connecting the first set of minimum points in the landscape via an edge of weight zero, a connected MST can be found which encompasses all the vertices in the graph. The individual subtrees starting at the set of minimum points then form the output of the algorithm.

4. **Erecting watersheds.** Functioning as an artificial divide between two adjacent catchment basins, watersheds provide the final segmentation lines of the process described. When to erect a watershed vs. letting two regions merge (by being overflowed from one side) is a question depending on how many segments are wanted, so no universal decision criterion exists. As watershed methods often produce over-segmented images, this is a very important point in implementing such a procedure.

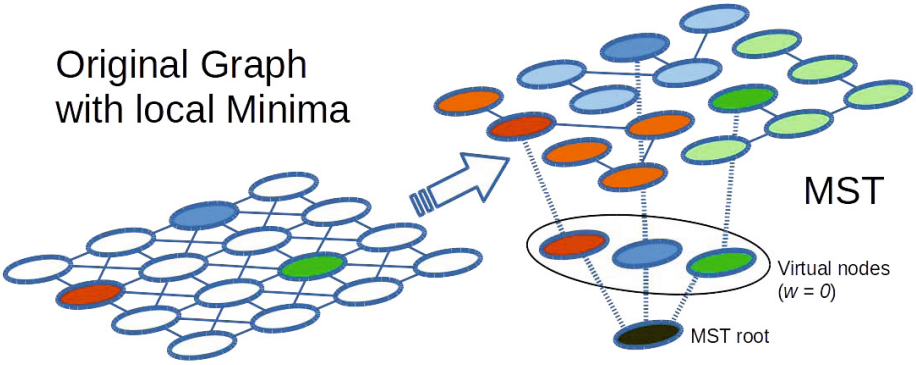

Fig. 1. MST view of finding catchment basins: According to [16] a minimum spanning tree can be computed using auxiliary (virtual) vertices. The weight of the edges represents a dissimilarity measure between pixels / voxels.

3.3 Region Merging

This section briefly describes a region merging algorithm which was published by [17]. This algorithm forms the basis of our own image segmentation implementation whose results will be discussed in a later section.

The original digital image can be interpreted as a directed, conservative weighted graph G, whose vertices V are the pixels themselves, where every (non-bordering) vertex has eight edges to its neighbouring nodes. Every edge $e \in E$ is weighted with the difference of the intensity values $i \in \{0 \dots 255\}$ of the pixels it connects. In this initial phase every pixel also constitutes its own region. To group similar pixels into a region R, where R is a subset of V, the following steps (written as pseudo algorithm) are needed:

1. Sort all edges $e \in E$ in ascending order.
2. For all edges e do:
   (a) Check if the pixels connected by e are already in the same region. If yes, continue to the next edge.
   (b) If not, check if there exists a boundary between the regions under observation. There exists a boundary if the minimum edge weight connecting those regions is greater than the minimum of the maximum internal MST-edges of the respective regions, where MST denotes the Minimum-Spanning Tree of the regions the pixels belong to.
   (c) If a boundary exists, continue to the next edge.
   (d) Else, merge the regions and update any pertinent properties (internal MST, avg color, gradients, etc.).

3.4 Segmentation Techniques Using Hybrid Approaches

Aside the rather traditional approaches mentioned earlier there has also been published some interesting work in methods using anterior knowledge about

object primitives, employing supervised learning etc. An example of this is the approach described in [18]. First, they propose extracting known and unknown objects from an image - this of course presupposes the existance of such knowledge and an efficient lookup possibility in an object database. Once those primitives are established, they compute object histograms over several different distances for each single object previously found. Thus they strive to derive an understanding about the global image structure from individual regions. For instance, if an object were a tree, and its histograms of surrounding objects would comprise (in ascending order of distance): 3 trees, 7 tress, 25 trees..., one could infer that the image depicted a forest. If, however, the histograms would show: 3 trees and 3 windows, 5 trees and 5 windows, 7 trees and 7 windows..., the image might rather depict an alley. Although in our work we are not yet concerned about object recognition or category discovery, this modern approach is rather similar to ours and we are looking forward to seeing further advancement in the area.

4 Experimental Setup

Our overall goal is to establish a software framework for graph extraction and graph analysis which is open source and accessible via a Web site. Specifically, as technology in Virtual Machines has improved dramatically since the late 2000s, it is now feasible to conduct such computations client side in a language like Javascript, which would have been fantastic only a few years back. The advantages to this approach are manifold: First, the possibility of running a low-cost infrastructure as scaling is done automatically by users providing their own computational power. Second, we have to store only the compressed graph structure (in JSON) along with some metadata concerning the algorithms and parameters used; storing the complete images is no longer required. Third, the processing is done faster because the additional computation time necessary in Javascript is now less than the time it takes to upload an image. Fourth, we can thus make use of the amazing visualization capabilities now offered for free by modern Javascript libraries; this will allow our users to immediately see and interact with the results of our computations.

4.1 Processing Pipeline

In extracting a graph from a natural image, we generally follow 4 consecutive steps whose specific implementation may be switched and whose input and output datastructures may vary depending on the chosen segmentation algorithm. Nevertheless, they form a logical flow which we intend to formalize in later versions of our framework in order to provide a standard procedure that user extensions can be plugged into.

1. **Image Preprocessing.** As a first step we may need to apply some preprocessing operations, e.x. the conversion to an intensity (grayscale) image or a background separation step. In any case both input and output of this step are images.

2. **Algorithmic Preprocessing.** There exist a wide variety of image segmentation algorithms, some of which are graph-based (like the region merging approach described earlier), while others would use clustering, compression etc. Therefore, the second step in our pipeline consists of providing the datastructures needed by the particular class of segmentation method. For the example case in this paper, we transform the image to an initial graph structure, with every pixel forming it's own region, and provide an adjacency list and edge list representation for further computations.
3. **Image Segmentation.** The core of our processing pipeline consists of the actual image segmentation step, which transforms the datastructures provided to it into a label map denoting each pixels affiliation to a region. In this step users should be able to choose among different classes and specific implementations of algorithms. In the future we also intend to give users the opportunity to implement and upload their own code, which will be injected into the pipeline.
4. **Graph extraction.** Based on the label map produced in the preceding step we can now extract the graph structure by first computing the region centroids followed by a Delaunay triangulation on the resulting set of vertices. Additionally, depending on the chosen segmentation algorithm and implementation, a representative feature vector will be stored for each region. This might include information like average color, gradients, or environment histograms, and in the future will be adaptable by the user as well.

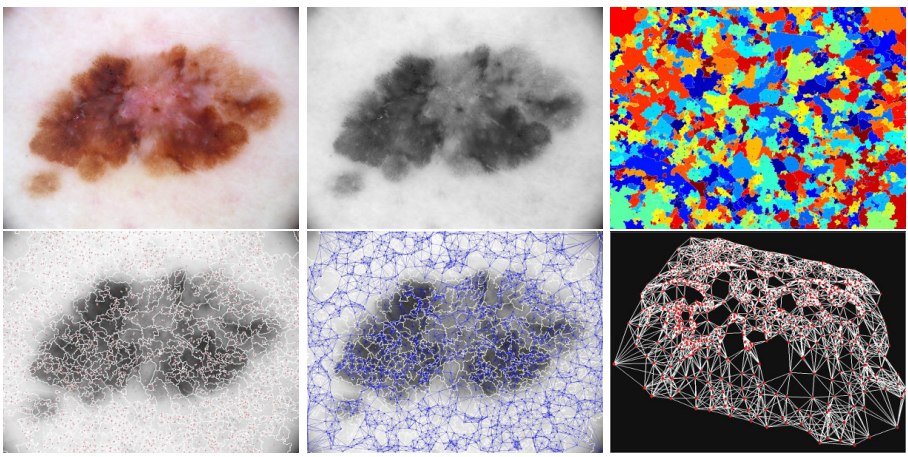

Fig. 2. A depiction of our computational pipeline: The input image (1) is transformed by a simple preprocessing step (2), then we segment the image using the Meyers 1994 watershed algorithm (Matlab implementation) [19] (3). Once the regions are obtained, centroids (4) as well as k-nearest neighbors (5) are computed and the graph is stored as a JSON datastructure visualized by the three.js Javascript framework (6).

5 Results

As we are building our graph extraction framework from the ground up, we have yet only implemented a single algorithm to demonstrate its feasability: We chose the Kruskal-based region merging algorithm described in [17] and implemented it in Javascript, adding two additional parameters to the algorithm: While the original paper only utilized k, which defined an input to the threshold computation above which two regions would be merged, we are also using s (size-threshold), the minimum size of pixels a region has to contain in order to be considered in the final graph construction phase, as well as m (max-merge-size) which gives the maximum amount of pixels a region may be grown to.

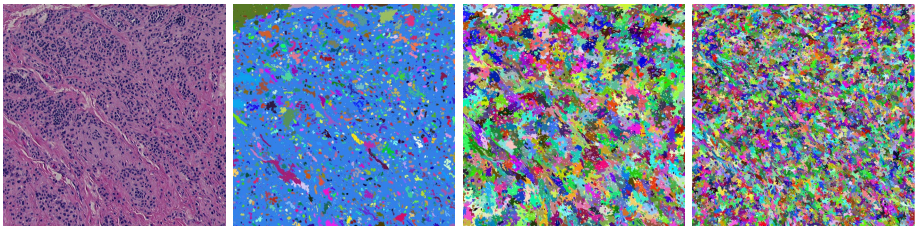

Fig. 3. Result of applying a Kruskal based region merging algorithm to an image of numerous small scale regular structures. (1) Input image, (2) Result with parameters $k = 1150, s = 0, m = \infty$, (3) Result with parameters $k = 150, s = 5, m = 500$, (4) Result with parameters $k = 50, s = 2, m = 150$.

In order to be able to compare our results, we performed a watershed-based segmentation plus graph extraction in Matlab as well. To that purpose, we chose a simple algorithm wich converts the RGB into an intensity image, performs a top hat filtering followed by a grey level threshold computation. It then converts the image to a binary matrix which the watershed is finally performed on. The algorithm only uses 1 parameter d to control its behavior - the size of the disk-shaped morphological structuring element that is used in the top hat filtering.

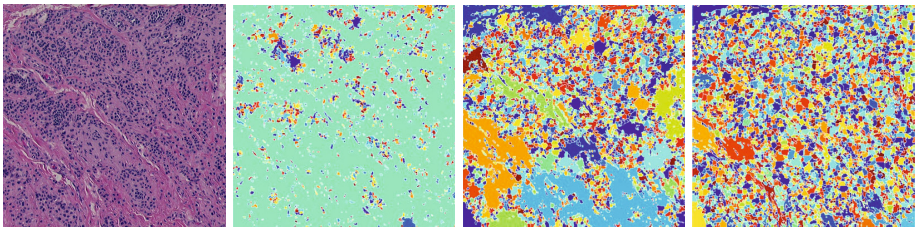

Fig. 4. Results of applying a Matlab Watershed algorithm to an image of numerous small scale regular structures. (1) Input image, (2) Result with parameter $d = 5$, (3) Result with parameter $d = 10$, (4) Result with parameter $d = 20$.

As we haven't implemented the watershed segmentation ourselves and there are diverse ways of doing this, we will not delve into details about the quality of the resulting images. However, we would like to compare the execution time of both algorithms for the different parameter settings as well as the resulting graph sizes, as those are important for any further computation. Our test system was equipped with a Core i5 Quad IvyBridge CPU, 8 GB of RAM and an SATA III SSD drive. No hardware acceleration was used in either case.

Table 1. Runtimes and graph sizes for different algorithms & parameter settings

Algorithm	k	s	m	d	**Nr.Vertices**	**Runtime in ms**
ML / Watershed				5	2,350	493
ML / Watershed				10	5,065	1,044
ML / Watershed				20	6,323	1,359
JS / Kruskal	1150	0	∞		3,952	3,178
JS / Kruskal	150	5	500		4,169	3,220
JS / Kruskal	50	2	150		13,916	3,863

Although at first glance it would seem that the Matlab based watershed algorithm clearly outperforms the Javascript based Kruskal segmentation, it is worth noting that the latter has only been in existence for about 2 weeks and no optimization has been performed on the code (see below). Moreover, since region merging depends on a sorted edge list for the original image graph, this base operation does not change with different parameters. Last, our Kruskal based method for the most extreme set of parameters produces a graph of about twice the number of vertices than the largest graph emitted by ML / Watershed.

6 Open Problems

Graph based analysis of image data on the Web (-browser) is still a novel topic. Consequently, there are many open issues to address, two of which will affect us in the immediate future.

- **Performance.** Even though our first results are already very promising and could not have been achieved only a few years ago, there is still a small gap in the performance of Javascript and highly optimized, Desktop-based compiled libraries. In order to address this issue, we propose to perform three improvements on our initial code base. First, our code currently uses generic Javascript data types, while typed datastructures allow significantly higher performance - this can be done by simple refactoring. Second, new technologies like asm.js would enable us to write low-level pieces of code in C and compile them to highly optimized Javascript. This requires special support by JS Virtual Machines, but has already shown the potential to

speed up code execution to within 2x the runtime of native compiled C code. Third, a shift to parallel implementations of our algorithms would provide the opportunity to outsource those operations to a GPU (e.x. via WebGL), which could reduce runtime to a small fraction of today's.
- **Quality of extracted graphs.** At this point we can already extract graph structures of different sizes and forms, but we lack a metric to judge if the obtained graph is suitable for further computation towards a given goal. As there seem to be no scientific graph analysis libraries available in JS today, we will have to implement our own in order to compare the results to those obtained by traditional image segmentation or manual diagnosis [20].

7 Conclusion and Future Work

Hot and promising topics for future research in knowledge discovery and data mining from natural image data is in the application of sophisticated topological methods and machine learning approaches, where natural images are seen as topographical landscapes, or map structures, similar to a terrain network [21]. On such landscapes autonomous multi-agents [17] [22], e.g. ant-robots [23], can leave markings on "interesting" areas, where such markings can be sensed by all robots and allow them to cover the unknown terrain without direct communication with each other, e.g. to discover anomalies, similarities or dissimilarities in images - exactly the aim of knowledge discovery and data mining. In the near future we will focus on the following issues:

- **Multistage processing.** To get even better results on small scale regular structures, it could be useful to perform several passes of our methods. For instance, background separation could be achieved with certain parameters in a first step. Afterwards one can apply the method again in order to find similar structures within the remaining boundaries.
- **Compression / reconstruction of images via topographic maps.** When a digital image is converted to a topographic map, the whole image information could theoretically stay complete, in which case the image can be fully restored afterwards. This might lead to a new approach in image compression.
- **Similarity measure on graph structures.** One major problem of taking a single digital image is measurement errors (artefacts) stemming from electronic fluctuations in the picture taking device. We could avoid these mistakes by taking pictures in short sequence and merging them in a meaningful way. Furthermore identifying similarities and differences between these images could help to improve the quality and stability of the resulting graphs, thus enabling us to get more reliable results from the data.
- **Extendable Web based research platform.** In order to make our platform valuable to a variety of researchers, we not only need to implement a range of algorithms ourselves, but enable our users to easily exchange their results or even upload their own code to test it on predefined images. This would be interesting from our perspective as storing different

$image - algorithm - parameter$ sets opens up the way to meaningful comparisons of results as well as to applying machine learning techniques on the whole processing pipeline. Moreover, it is also desirable to our users as they could use our platform as a publishing service, making their research accessible / reproducible via simple bookmarking.

References

1. Holzinger, A., Dehmer, M., Jurisica, I.: Knowledge discovery and interactive data mining in bioinformatics state-of-the-art, future challenges and research directions. BMC Bioinformatics 15(suppl. 6), S1 (2014)
2. Bunke, H.: Graph-based tools for data mining and machine learning. In: Perner, P., Rosenfeld, A. (eds.) MLDM 2003. LNCS, vol. 2734, pp. 7–19. Springer, Heidelberg (2003)
3. Strogatz, S.: Exploring complex networks. Nature 410, 268–276 (2001)
4. Dehmer, M., Emmert-Streib, F., Mehler, A.: Towards an Information Theory of Complex Networks: Statistical Methods and Applications. Birkhaeuser, Boston (2011)
5. Holzinger, A.: On topological data mining. In: Holzinger, A., Jurisica, I. (eds.) Interactive Knowledge Discovery and Data Mining in Biomedical Informatics. LNCS, vol. 8401, pp. 331–356. Springer, Heidelberg (2014)
6. Holzinger, K., Palade, V., Rabadan, R., Holzinger, A.: Darwin or lamarck? future challenges in evolutionary algorithms for knowledge discovery and data mining. In: Holzinger, A., Jurisica, I. (eds.) Interactive Knowledge Discovery and Data Mining in Biomedical Informatics. LNCS, vol. 8401, pp. 35–56. Springer, Heidelberg (2014)
7. Holzinger, A., Blanchard, D., Bloice, M., Holzinger, K., Palade, V., Rabadan, R.: Darwin, lamarck, or baldwin: Applying evolutionary algorithms to machine learning techniques. In: The 2014 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2014). IEEE (in print, 2014)
8. Holzinger, A., Jurisica, I.: Knowledge discovery and data mining in biomedical informatics: The future is in integrative, interactive machine learning solutions. In: Holzinger, A., Jurisica, I. (eds.) Knowledge Discovery and Data Mining. LNCS, vol. 8401, pp. 1–18. Springer, Heidelberg (2014)
9. Otasek, D., Pastrello, C., Holzinger, A., Jurisica, I.: Visual data mining: Effective exploration of the biological universe. In: Holzinger, A., Jurisica, I. (eds.) Knowledge Discovery and Data Mining. LNCS, vol. 8401, pp. 19–33. Springer, Heidelberg (2014)
10. Albert, R., Barabási, A.L.: Statistical mechanics of complex networks. Reviews of Modern Physics 74, 47–97 (2002)
11. Makrogiannis, S., Economou, G., Fotopoulos, S., Bourbakis, N.G.: Segmentation of color images using multiscale clustering and graph theoretic region synthesis. IEEE Transactions on Systems Man and Cybernetics Part A: Systems and Humans 35, 224–238 (2005)
12. Kropatsch, W.G., Burge, M., Glantz, R.: Graphs in image analysis. In: Kropatsch, W.G., Bischof, H. (eds.) Digital Image Analysis, pp. 179–197. Springer, New York (2001)
13. Caselles, V., Coll, B., Morel, J.M.: Topographic maps and local contrast changes in natural images. International Journal of Computer Vision 33, 5–27 (1999)

14. Ahammer, H., Kröpfl, J.M., Hackl, C., Sedivy, R.: Image statistics and data mining of anal intraepithelial neoplasia. Pattern Recognition Letters 29, 2189–2196 (2008)
15. Vincent, L., Soille, P.: Watersheds in digital spaces: an efficient algorithm based on immersion simulations. IEEE Transactions on Pattern Analysis and Machine Intelligence 13, 583–598 (1991)
16. Straehle, C., Peter, S., Köthe, U., Hamprecht, F.A.: K-smallest spanning tree segmentations. In: Weickert, J., Hein, M., Schiele, B. (eds.) GCPR 2013. LNCS, vol. 8142, pp. 375–384. Springer, Heidelberg (2013)
17. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. International Journal of Computer Vision 59, 167–181 (2004)
18. Lee, Y.J., Grauman, K.: Object-graphs for context-aware visual category discovery. IEEE Transactions on Pattern Analysis and Machine Intelligence 34, 346–358 (2012)
19. Meyer, F.: Topographic distance and watershed lines. Signal Processing 38, 113–125 (1994)
20. Holzinger, A., Malle, B., Bloice, M., Wiltgen, M., Ferri, M., Stanganelli, I., Hofmann-Wellenhof, R.: On the generation of point cloud data sets: Step one in the knowledge discovery process. In: Holzinger, A., Jurisica, I. (eds.) Knowledge Discovery and Data Mining. LNCS, vol. 8401, pp. 57–80. Springer, Heidelberg (2014)
21. Preuß, M., Dehmer, M., Pickl, S., Holzinger, A.: On terrain coverage optimization by using a network approach for universal graph-based data mining and knowledge discovery. In: Slezak, D., Peters, J.F., Ah-Hwee, T., Schwabe, L. (eds.) Brain Informatics and Health. LNCS (LNAI), vol. 8609, pp. 569–578. Springer, Heidelberg (2014)
22. Olfati-Saber, R., Fax, J.A., Murray, R.M.: Consensus and cooperation in networked multi-agent systems. Proceedings of the IEEE 95, 215–233 (2007)
23. Wagner, I., Bruckstein, A.: From ants to a(ge)nts: A special issue on ant-robotics. Annals of Mathematics and Artificial Intelligence 31, 1–5 (2001)