Seventh Hungarian Conference on Computer Graphics and Geometry, Budapest, 2014

3D mesh generation from aerial LiDAR point cloud data

Péter Polcz and Csaba Benedek

Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI)
polcz.peter@sztaki.mta.hu, benedek.csaba@sztaki.mta.hu
http://web.eee.sztaki.hu/remotesensing

Abstract

Three dimensional urban scene modelling became important issue in the last few years. Beside visual experience, 3D city modelling has gained a significant function in diverse analysing tasks, however the amount of data requires a high level of automation of model generation. In this work, we introduce an automatic and robust algorithm which produces detailed 3D virtual city models by analysing high resolution airborne LiDAR point clouds. Using the idea of the surface normal based roof segmentation we have designed a procedure, which takes into account the boundaries of each roof segment, so that the adjacent segments connect without gaps. We have developed an algorithm to detect 3D edge lines of the rooftops. Since the applied triangulation methods operate on the whole convex hull of the input points, hollow outer parts of the roof segments are filled in with false triangles. To solve this problem, we have proposed a method using a Markov Random Field, in which we filter out the incorrect triangles lying on the concave parts.

Categories and Subject Descriptors (according to ACM CCS): I.4.5 [Computer vision]: Reconstruction

1. Introduction

In the last decade, LiDAR (Light Detection and Ranging) has been widely used in various remote sensing application fields. LiDAR is an optical remote sensing technology that can measure the distance of targets from the scanner by illuminating the target with laser light and analysing the backscattered light, therefore a such a laser scanner yields a 3D point cloud representing the objects around the scanner.

A specific type of these sensors can be mounted on airplanes, and the provided scans are appropriate for creating digital terrain models (DTM) and digital elevation models (DEM). These models are efficient and detailed descriptions of fields, valleys, mountains or other desert areas. These irregular, rough and mountainous terrain types cannot be represented as a set of regular shapes.

On the other hand, in case of cities or other urban settlements polygon reconstruction constitutes another alternative solution for modelling. The main targets of the reconstruction are the buildings having regular geometrical shapes introducing the possibility to approximate them with several three-dimensional polygons. Worldwide projects (Google maps 3D, Nokia maps) are devoted to this topic.

As input, we have used high resolution LiDAR records of Budapest city center, which have been provided by Infoterra Astrium GEO-Information Services Hungary.

In this paper we intend to present our approaches of aerial point cloud processing, three-dimensional city reconstruction and urban scene modelling.

2. Previous Work

This research domain has considerably progressed during the last decade. Many of computer vision researchers have developed new techniques and algorithms in order to create not only realistic but also simple[1] city-models at the same time. Regarding the latest publications, significant results have been encountered by Lafarge et al. [1, 2], Zhou et al. [3–5], Huang et al. [6, 7] and Verma et al. [8]. Zhou's approach consists in geometrical and topological corrections of an initial mesh on the basis of local observations of the buildings' orientation. Whereas Lafarge and Huang defined geometric 3D primitives to fit them to the different building types and rooftop shapes appearing in the point cloud.

[1] in the means of reduced number of facets

Lafarge et al. [1, 2] also handled non-planar primitives as cylinders, spheres and cones.

Huang et al. [6, 7] created complete roof models (composed by planar primitives) and attempted to fit them to the cloud regions classified as buildings using different statistical methods (in particular likelihood function maximization). After a geometrical adjustment, the primitives were "merged" into a plausible model.

Verma et al. [8] also used a statistical approach, by building a dual graph from the roof segments. However, this technique only worked for planar roof models.

3. A brief description of our proposed reconstruction algorithm

The workflow includes four steps, from which the first three steps are illustrated in Figure 1. First, the point cloud is classified using an unsupervised method based on the work of Börcs and Horváth [9] and the method introduced by Lafarge et al. [1, 2] and Zhou [5], in which the algorithm distinguishes four different classes: ground, building, vegetation and clutter. Then the point cloud regions classified as buildings are divided into several parts in order to reduce the complexity of the further steps. Each part of the cloud will contain a reduced number of points belonging to a single complex rooftop.

Secondly, the proposed algorithm approximates each rooftop by planar shaped faces. These planar roof segments are extracted by a robust method detailed in Section 4. The points of a roof component determine a plane, which is calculated through minimizing the sum of squared distances of the points from the plane.

The third step is described in Section 5, which consists in generating a 3D skeleton model for each building block by detecting the roof's edge lines. After triangulating the endpoints of the edge lines we retain several, approximately planar shaped polygon meshes which will form together a 3D building model.

The last step constitutes the main part of our contribution, in which we introduce a new method for concave triangle mesh generation which solves the problem of concave shaped roof segments.

4. Roof segmentation

In this section separate planar roof segments on the basis of their orientation. First of all, we estimate a local surface normal at every point of the roof cloud using the Point Cloud Library's [10] implementation of Moving Least Squares (MLS) algorithm (Figure 2 - left). Since we know every point's normal, we apply a clustering algorithm to detect the representative directions in which the planar roof components face (black vectors in Figure 2 - right). These

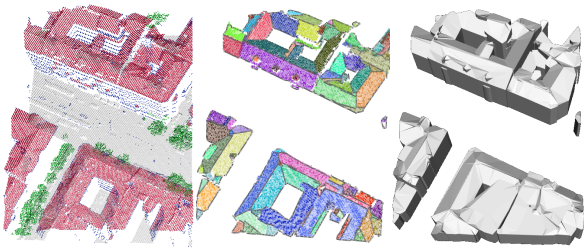

Figure 1: The first three steps of the proposed method: initial classification (left), roof segmentation and edge detection (middle), triangle mesh generated from endpoints of the edge lines.

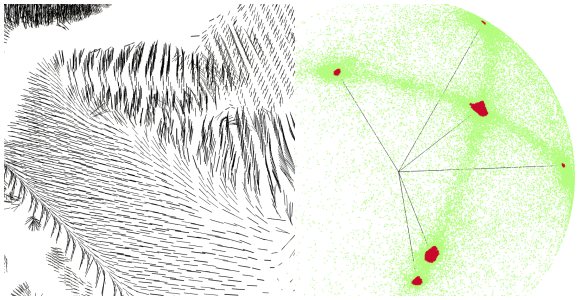

Figure 2: Illustration of our surface normal based clustering. First the normals are calculated using an MLS algorithm (left), after that the normals are translated into the origin. The second image illustrates the endpoints of the normal vectors, from which dense regions are extracted and than clustered.

few directions will represent separate clusters with different labels. In the following, every point will be assigned an appropriate label (i.e. color), depending the point's normal. As a result, the points of every roof segment having similar orientation will be given the same label. Afterwards, a region-growing is applied on the cloud knowing the labels that the roof points belong to. Since the segmentation produces a slightly noisy label mask, we adopt a further smoothing step, which uses the K-nearest neighbors smoothing algorithm. At the end we retain the final labeling, in which every planar continuous roof component are distinguished by a unique segment label, and then we will be ready to perform the polygon approximation for every roof segment.

5. 3D edge detection and triangulation

3D edge lines are detected in two different ways. Let us call *inner edges* those, which lie alongside the connection of two neighboring faces of the roof (ridges). These lines are identified as the intersections of the respective neighboring planar roof components.

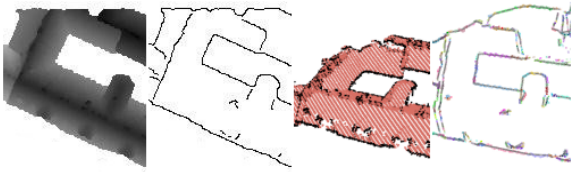

Figure 3: Outer edge detection's assembly - projection (z-image) - edge detection - elevation into the 3D space - segment fitting.

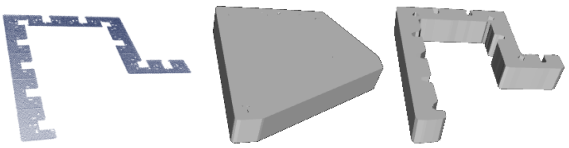

Figure 4: Concave problem - points of a concave roof segment (left) - triangulation on the whole convex hull (middle) - triangle mesh generated by our method (right)

On the other hand, we call *outer edges* the lines, which constitute the outer boundaries of the rooftops (eaves). Since in the airborne LiDAR point clouds, we usually have no reflection from the vertical walls, outer edges are calculated through image processing techniques, as illustrated in Figure 3.

First the roof cloud is projected onto the xy horizontal plane so that each pixel will get the respective 3D point's z elevation value, as its grayscale color value. Henceforth, we call this projected image as z-image. After adopting an edge detection algorithm on the z-image, we retain an edge image in which high elevation differences are highlighted. Using the z-image and the edge image we elevate the edge points into 3D, and we fit 3D lines to the 3D edge points. The generated 3D lines constitute the *outer* edge lines of the rooftops.

The rooftops' planar faces are generated by triangulating the endpoints of the edge lines. Vertical outer walls are produced using the outer triangle sides of the previously generated triangle meshes.

6. Concave triangle mesh generation

Concave shaped roof segments appear frequently, however several well established triangulation methods, such as the used Delanuay triangulation[2] provided by CGAL [12], generate triangle meshes on the whole convex hull of the given points[3]. Consequently, as shown in Figure 4 (middle), important architectural features of the building may be filled in with false roof components. Therefore we designed a procedure in which triangles lying on the concave parts of the Delaunay convex mesh will be erased preserving smooth boundaries in the final concave mesh. The procedure is based on a probabilistic graphical model.

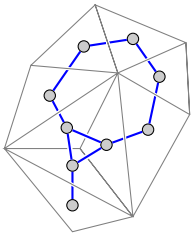

Figure 5: 3D triangle mesh and its corresponding undirected triangulation graph

According to Gansner, Hu and Kobourov [13] a triangle mesh is defined by the included triangles (S) and the neighborhood connections ($\mathcal{N}$) between them, hence it can be modeled as an undirected graph (Figure 5) where each triangle is considered as a separate vertex ($s \in S$), and each neighborhood connection as an undirected link ($\{s,r\} \in \mathcal{N}$) between two vertices ($s,r \in S$) corresponding to the neighboring triangles. Furthermore, some of the triangles in the mesh need to be deleted because they are lying on the concave parts of the roof segment. As a consequence, we have to assign a random variable (Ω_s) to each vertex that marks the fact whether the corresponding triangle needs to be eliminated or not. After classification, we call a given triangle as *relevant triangle*, if it should be kept in the final mesh. These Ω_s variables are defined on the set of vertices (S), therefore they form a $\Omega = (\Omega_s)_{s \in S}$ random field with respect to $\mathcal{N}$.

6.1. Markov property

In our case *planar graph* models can also be used, since the considered meshes are open[4] (i.e. they have at least three triangles having neighbors fewer then three), and do not contain any wholes. *Planar graphs*[5] can be drawn on the plane, so that their edges intersect only at their endpoints. It is convenient to use them, since they satisfy the below constraints[6]:

- we have an upper limit for the number of edges: $e \leq 3v - 6$, where e is the number of edges and v is the number of vertices.
- the maximum number of vertices of a fully connected (complete[7]) sub-graph is 4.

With reference to Gansner, Hu and Kobourov [13] (Lemma 1. on pg. 5.) we cannot draw on the plane four triangles

[2] described in detail by Gallier et al. [11] in Section 8.3 *Delaunay Triangulations*

[3] "there is an intimate relationship between convex hulls and Delaunay triangulations", pronounced by Gallier et at. [11] in Section 8.4 *Delaunay Triangulations and Convex Hulls*

[4] they are open 2-manifolds (Smith et al. [14] on pg. 14.)

[5] see definition given by Balakrishnan et al. [15] (Definition 8.2.1 on pg. 175.)

[6] see theorems and their consequences formulated by Balakrishnan et al. [15] in Section 8.3 *Euler Formula and Its Consequences*

[7] see definition given by Balakrishnan et al. [15] (Definition 1.2.11)

which are all connected to each other, therefore the maximum number of vertices of a complete sub-graph is three. Accordingly, the vertices of the graph are usually connected with a reduced number of other vertices especially in their close proximity (Markov property). Furthermore, we presume that every triangle's label is conditionally independent of any other non-adjacent node's label, given the labels of all neighboring triangles. With regard to Stan Z. Li et al. [16], Ω is said to be a Markov Random Field (MRF) on S wrt. $\mathcal{N}$.

6.2. Prior and data model

MRFs are able to simultaneously embed a data model, reflecting the knowledge on the observation; and prior constraints, such as spatial smoothness of the solution. As for the ***prior model***, we used the following energy function:

$$E_s = \sum_{\{s,r\}\in\mathcal{N}} V(\omega_s, \omega_r)$$

where V implements a smoothing constraints, using the Kronecker delta: $V(\omega_s,\omega_r) = \delta(\omega_s = \omega_r)$. In case of a particular $s \in S$ triangle, our ***data model*** uses the following descriptors:

n_s : number of points projected into s
A_s : area of s
φ_s, ϑ_s : two arbitrary angles of s

Using these measures we will generate a single x_s fitness value for each triangle s, so that x_s will be approximately proportional with the likelihood of the fact that s constitutes a relevant element in the concave triangulation, hence it should not be deleted from the final mesh. As for the first feature, we calculate the density of the projected points in each triangle ($\frac{n_s}{A_s}$), and we divide the calculated value by a

$$\mathcal{K} = \max_{s_i \in S} \frac{n_{s_i}}{A_{s_i}}$$

normalization coefficient, so that we obtain a density feature in the interval $[0,1]$. Let us introduce the following notation for this descriptor (ρ stands for density):

$$\rho_s = \frac{1}{\mathcal{K}} \cdot \frac{n_s}{A_s} \in [0,1] \qquad (1)$$

Next, we use our observation that bays (i.e. internal regions of the mesh which should be likely eliminated) contain mainly acute-angled triangles, while micro concavities on the boundaries of the open mesh consist of long and thin obtuse triangles. As a consequence, we introduced a so-called *angle cost* that measures how much a given triangle is obtuse. Let us define *angle cost* as the product of each angle's cosine values in the triangle.

$$\alpha_s = \big(\cos(\varphi_s)\cos(\vartheta_s)\cos(\pi-\varphi_s-\vartheta_s)+1\big)\cdot\frac{1}{1.125}$$

The angle cost gives its maximum value, if the triangle is equilateral ($\varphi = \vartheta = \pi/3$). Otherwise, the more a triangle is acute, the more its angle cost tents to 0.

Finally, a joint fitness value, i.e. a pseudo probability is defined as the product of ρ and $(1-\alpha)$, which indicates us, whether a given triangle is a relevant element of the mesh. These $x_s = \rho_s \cdot (1-\alpha_s)$ values will form our ***observed feature layer*** (Figure 6).

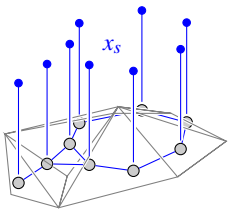

Figure 6: dependency graph, in which the filled dots stand for the x_s observed feature layer

However, during our experiments we perceived that excluding triangles just by their low x_s values using a given hard threshold can cause several false positive/negative triangles. In other words, the designed feature (x_s) is not enough in itself to decide whether a triangle belongs to the concave parts of the mesh or not. To overcome this limitation, we started to compare each triangle's class label with labels in its neighborhood, hence we took the advantage of the *prior model*. Just for illustration (Figure 7), let us color relevant triangles white and triangles able to be skipped gray. This color can also be interpreted as a label marking that the corresponding triangle is *relevant* or *removable*. If two or three neighboring triangles are gray the actual triangle is likely to be gray too, especially when the nearby triangles have larger area then the actual one.

For the $d_s(\omega_s) = d(x_s \mid \omega_s)$ *data cost* we have chosen the following functions:

$$d(x_s \mid \Omega_s = \text{relevant}) = f(1-x_s)$$
$$d(x_s \mid \Omega_s = \text{removable}) = f(x_s) \qquad \forall\, s \in S$$

where $f(x)$ is the sigmoid function, operating as a soft threshold:

$$f(x) = \frac{1}{1+e^{-n(x-x_0)}}\cdot(b-a)+a, \qquad f:[0,1]\longrightarrow[a,b]$$

with gradient $n = 20$, shift $x_0 = 0.5$, offset and scale $(a,b) = (0.2,2)$. Thereafter the *data model* of the MRF has the following form:

$$p(x_s \mid \omega_s) = e^{-d(x_s \mid \omega_s)}$$

Note that the above quantities define pseudo probabilities, since they are unnormalized.

Using our defined *prior* and *data model*, we can determine the ***posterior*** likelihood $P(\omega|x)$ of every possible global labeling over the triangle-graph, and we have no other tasks but choosing the most probable global labeling that will point out the desired (*relevant*) triangles. To conclude, we have optimized the following energy function using graph cuts based optimization technique developed by

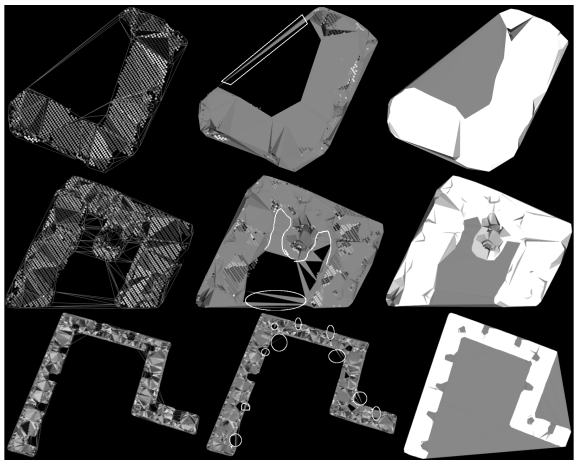

Figure 7: Removing triangles that do not belong to the concave hull. The first two columns demonstrate a hard threshold of the x_s feature, without the MRF smoothing constraint: we can observe several false triangles in the resulting meshes (2nd column). The 3rd column shows the optimal label mask where maximum probability is met. In the first column it can also be observed through coloring how the points are associated to the corresponding triangles.

Olga Veksler, using the libraries provided by Yuri Boykov and Vladimir Kolmogorov [17–20]:

$$\omega_{opt} = \arg\min_{\omega \in \Gamma} \left(\sum_{s \in S} d(x_s \mid \omega_s) + \lambda \sum_{\{s,r\} \in \mathcal{N}_s} \delta(\omega_s, \omega_r) \right)$$

The results are illustrated in Figure 7, which shows smooth features at the roof segments' boundaries in the same way false triangles are eliminated.

7. Results

As Figure 8 illustrate, the algorithm can be applied for a wide range of building types even though it solely estimates the geometry of objects by several planar elements (polygons). We have reconstructed city sites featuring urban civil apartment houses (Figures 1 and 8 - city site), buildings with complex architectural roof models (Figure 8 - Market Hall and BUTE K-building), large concave blocks of flat (Figure 4). The algorithm was tested on different point clouds containing together about three million points, covering an area of $102,000\ m^2$. The aggregate number of the generated triangles is about $200,000$ without triangles of the outer walls. See Table 1 for the details.

8. Future plans and conclusion

Our work's primary objective was to design an automatic and robust method to process aerial LiDAR data and produce

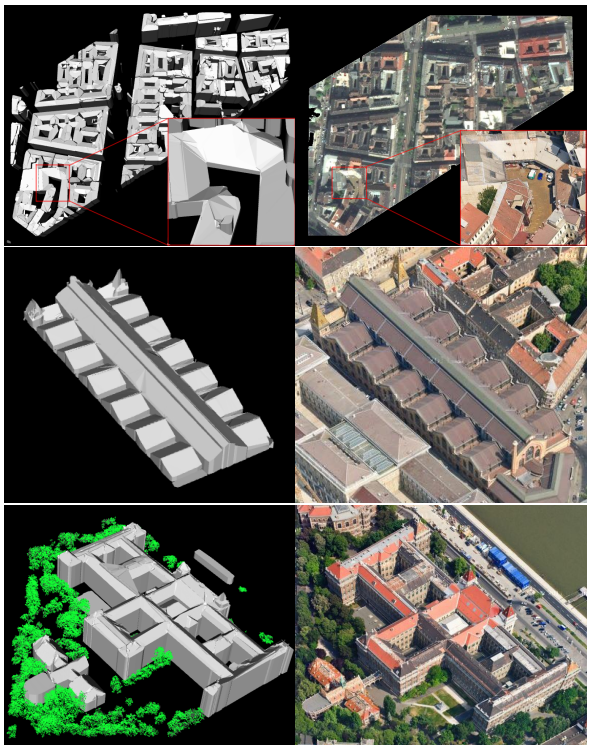

Figure 8: Our polygon reconstruction results (left), and the reference aerial photos (right) of various landmarks of Budapest. Civil apartment houses - Budapest's site between Mária St., Nap St., Futó St. and Baross St. (top), Vásárcsarnok Market, Vámház körút (middle), Budapest University of Technology and Economics (BUTE) - central building (bottom)

three-dimensional geometric models from them. The obtained three-dimensional models will be compared with optical images taken from the space in different times, analysing the possibilities of adaptive texturing and change detection.

9. Acknowledgement

We thank Infoterra Astrium GEO-Information Services ©, who gave us the permission to test our system on their aerial LiDAR records of Budapest. This work was founded by the Government of Hungary through a European Space Agency (ESA) Contract under the Plan for European Cooperating States (PECS), and by the Hungarian Research Fund (OTKA #101598).

References

1. Florent Lafarge and Clément Mallet. Creating large-scale city models from 3d-point clouds: A robust approach with hybrid representation. *International Journal of Computer Vision*, 99(1):69–85, August 2012.

city site	nr. of points	nr. of roof points	nr. of triangles[1]	$\frac{\text{nr. roof points}}{\text{nr triangles}}$	area (m^2)	proc. time	in Figure
Móricz sqr.	147,120	69,643	3,950	17.63	7,956	$\sim$ 2 min	1
Baross str.	1,927,656	948,461	160,684	5.90	35,954	$\sim$ 20 min	8 (top)
Market Hall	151,319	151,319	6,307	23.99	9,000	$\sim$ 1 min	8 (middle)
BUTE	875,106	362,024	38,544	9.39	49,163	$\sim$ 10 min	8 (bottom)

Table 1: Quantitive results of our mesh generator algorithm.
[1] without triangles of outer walls

2. Florent Lafarge and Clément Mallet. Building large urban environments from unstructured point data. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1068–1075, Barcelona, Spain, 2011.
3. Qian-Yi Zhou and Ulrich Neumann. Complete residential urban area reconstruction from dense aerial LIDAR point clouds. *Graphical Models*, 75(3):118–125, 2013.
4. Qian-Yi Zhou and Ulrich Neumann. Modeling residential urban areas from dense aerial LIDAR point clouds. In *Proceedings of the First international conference on Computational Visual Media*, CVM'12, pages 91–98, Berlin, Heidelberg, 2012. Springer-Verlag.
5. Qian-Yi Zhou. *3D urban modeling from city-scale aerial LIDAR data*. Phd thesis, Faculty of the Graduade School University of Southern California, 2012.
6. Hai Huang, Claus Brenner, and Monika Sester. 3d building roof reconstruction from point clouds via generative models. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '11, pages 16–24, New York, NY, USA, 2011. ACM.
7. Hai Huang, Claus Brenner, and Monika Sester. A generative statistical approach to automatic 3d building roof reconstruction from laser scanning data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 79(0):29–43, May 2013.
8. Vivek Verma, Rakesh Kumar, and Stephen Hsu. 3d building detection and modeling from aerial LIDAR data. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2 of *CVPR '06*, pages 2213–2220, Washington, DC, USA, 2006. IEEE Computer Society.
9. Attila Börcs and Csaba Horváth. Városi környezet automatikus analízise és rekonstrukciója légi LiDAR mérések alapján, TDK dolgozat, Pázmány Péter Katolikus Egyetem Információs Technológiai Kar. 2011.
10. Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *International Conference on Robotics and Automation*, Shanghai, China, 2011.
11. Jean Gallier. Notes on Convex Sets, Polytopes, Polyhedra Combinatorial Topology, Voronoi Diagrams and Delaunay Triangulations. Rapport de recherche RR-6379, INRIA, 2007.
12. CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.
13. Emden R. Gansner, Yifan Hu, and Stephen G. Kobourov. On touching triangle graphs. *CoRR*, abs/1001.2862, 2010.
14. Colin Smith. *On vertex-vertex systems and their use in geometric and biological modelling*. PhD thesis, Calgary, Alta., Canada, Canada, 2006. AAINR19574.
15. R. Balakrishnan and K. Ranganathan. *A Textbook of Graph Theory*. Universitext - Springer-Verlag. Springer, 2012.
16. Stan Z. Li. *Markov Random Field Modeling in Image Analysis*. Springer Publishing Company, Incorporated, 3rd edition, 2009.
17. Richard Szeliski, Ramin Zabih, Daniel Scharstein, Olga Veksler, Aseem Agarwala, and Carsten Rother. A comparative study of energy minimization methods for markov random fields. In *In ECCV*, pages 16–29, 2006.
18. Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, November 2001.
19. Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:65–81, 2004.
20. Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9):1124–1137, sep 2004.