A Parametric Algorithm for Skyline Extraction

Mehdi Ayadi, Loreta Suta, Mihaela Scuturici, Serge Miguet, Chokri Ben Amar

A parametric algorithm for skyline extraction

Mehdi Ayadi[1,2], Loreta Suta[1], Mihaela Scuturici[1],
Serge Miguet[1], and Chokri Ben Amar[2]

[1] University of Lyon, CNRS
University Lyon 2, LIRIS, UMR5205
F-69676, France
[2] University of Sfax, ENIS
REGIM-Lab: REsearch Groups in Intelligent Machines
BP 1173, Sfax, 3038, Tunisia
{Mehdi.Ayadi,Mihaela.Scuturici,Serge.Miguet}@univ-lyon2.fr
loradrian@gmail.com
chokri.benamar@ieee.org

Abstract. This paper is dedicated to the problem of automatic skyline extraction in digital images. The study is motivated by the needs, expressed by urbanists, to describe in terms of geometrical features, the global shape created by man-made buildings in urban areas. Skyline extraction has been widely studied for navigation of Unmanned Aerial Vehicles (drones) or for geolocalization, both in natural and urban contexts. In most of these studies, the skyline is defined by the limit between sky and ground objects, and can thus be resumed to the sky segmentation problem in images. In our context, we need a more generic definition of skyline, which makes its extraction more complex and even variable. The skyline can be extracted for different depths, depending on the interest of the user (far horizon, intermediate buildings, near constructions, ...), and thus requires a human interaction. The main steps of our method are as follows: we use a Canny filter to extract edges and allow the user to interact with filter's parameters. With a high sensitivity, all the edges will be detected, whereas with lower values, only most contrasted contours will be kept by the filter. From the obtained edge map, an upper envelope is extracted, which is a disconnected approximation of the skyline. A graph is then constructed and a shortest path algorithm is used to link discontinuities. Our approach has been tested on several public domain urban and natural databases, and have proven to give better results that previously published methods.

1 Introduction

This work is part of interdisciplinary projects funded by the French National Research Agency (ANR-12-VBDU-008-Skyline) and by the Excellence Laboratory "Urban World Intelligence" (Labex IMU). These projects are dedicated to the study, in a very interdisciplinary way, of the immaterial "skyline" object. As mentioned in [2], a city's skyline is defined as *the unique fingerprint of a city* and it *abstracts a city's identity in terms of its spatial, historical, social, cultural*

and economic structures over time. One of the tasks of the projects is devoted to the subjective evaluation of the reception of the urban landscape by human subjects. Several pictures of the city are presented to people, that are then asked to evaluate the perceived skyline by marks ranging between 0 and 10, for different continuous variables: ugly-beautiful, disturbing-reassuring, messy-ordered,... The objective of our work is to measure the geometrical information from the skyline and to extract objective variables, that can be related to these subjective evaluations. The tool could be used for example to help urban planners to predict the most likely perception of different possible city's evolutions or construction scenarios. We therefore need a tool for automatically extracting the skyline in images, which is the main topic of this paper.

We found two possible operational definitions of the skyline :

1. The one-dimensional contour that represents the boundary between the sky and the ground objects [1]
2. The artificial horizon that a city's overall structure creates [3]

As will be illustrated in section 2, several papers are related to this problem of automatic skyline extraction. Most of them use the first definition, that resumes the problem of skyline extraction to the problem of sky segmentation: the skyline can be defined as the frontier of the sky region. Skyline extraction can be used in several applications: from geo-localization, path planning and obstacle detection to aerial and ground vehicles guiding.

In our work, we will rather use the second definition, which is more general, but also more versatile: As illustrated in figure 1a, we want our algorithm to be able to extract both the background skyline, including mountains, but also the foreground skyline, delimiting the silhouette of man-made buildings. We need thus a parametric, semi-automatic algorithm that allows the user to extract his own desired skyline, or a set of different users to define a range of valid skylines.

The paper is organized as follows: Section 2 presents the state of the art on skyline detection methods, which, according to definition 1, are considered as an image segmentation problem. They can be divided into region-based and edge-based approaches. Section 3 describes then our parametric skyline extraction algorithm based on low level image processing that allows to extract the upper envelope, which is a disconnected approximation of the skyline, and a graph-based modelization of the picture which allows to connect the skyline help to the execution of a shortest path algorithm.
We tested our algorithm on several databases of real images in various weather conditions and for several kinds of urban and natural landscapes dominated by man made buildings or including natural objects. Detailed steps and experimental results are explained in section 4. Finally, section 5 concludes our paper and presents our future works.

[3] See Wikipedia : https://en.wikipedia.org/wiki/Skyline

2 State of the art on skyline detection

In the last few years, there has been an increasing interest for finding algorithms and techniques to automatically extract the skyline in digital pictures. Skyline extraction is only a first step in a whole chain where the specific shape of the horizon can act as a signature for geo-spatial localization. In this paper, we focus on the first step of skyline extraction. Further works, using this skyline for augmented reality purposes are in progress (see section 5). For this extraction task, previous works are clearly classified as image segmentation problems that can be handled with common: *region-based* or *contour-based* approaches.

2.1 region based methods

In [3], a region based method is proposed where the algorithm first tries to automatically extract the sky region using a region growing method based on the luminosity's gradient. From the input image, 10 sub-windows are chosen at the top region of a video image. For each of these sub-windows, the average of intensity is computed. The minimum of the 10 obtained values is then be used as a threshold. Starting from the top of image, and for each column, first pixels whose intensities are lower than this threshold are considered as skyline's pixels. Otherwise, skyline extraction might fail due to the presence of clouds or noise in the image. Added to that, the extracted skyline is non continuous, as in [4], due to the non connection of skyline pixels in neighboring columns.

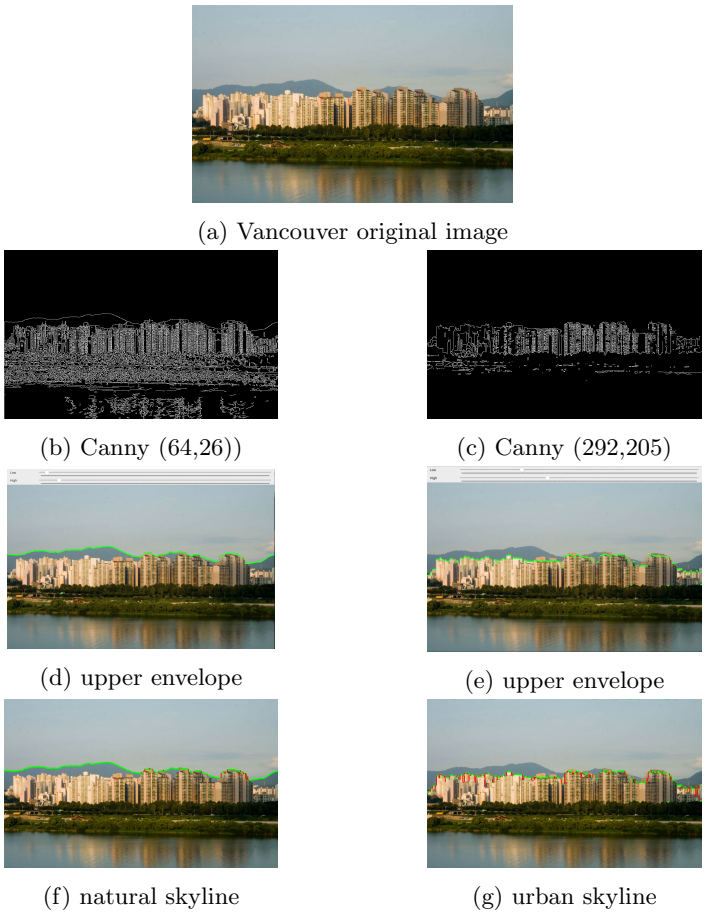
In another approach, [5], a neural network is used to assign a score in [0..1], to each pixel. Pixels that belong to the sky must have high score. However, in some cases, where clouds are present, the algorithm fails and increases the ratio of false positives.

In [6] or [7], authors propose to use machine learning, namely Support Vector Machine (SVM), to segment the image and extract the skyline. Used descriptors are essentially based on the color, static features and location information of edges. This step is followed by a dynamic programming step to link discontinuities. However, this approach allows to extract only the background skyline. In fact, as illustrated in figure 1a, images can present multi-level depth skylines. Added to that, such methods require a lot of computing power and memory resources

2.2 edge based methods

In [8], an edge detection step is applied on the luminance component to detect edge pixels and construct an edge map. This step is followed by an edge map construction stage where a dynamic programming algorithm allows to find paths between two horizontally consecutive disconnected vertices. First, exploration is permitted only on pixel's right area. This means that concavities or convexities, due to perspective problems when taking a photo or to original architectures, are not taken into consideration. Second, when a path cannot be found between two consecutive edge points separated with a vertical distance greater than a

predefined threshold, a linear interpolation is done. This does not reflect in all cases the real shape of the scene. Finally, cost of links between two vertices is calculated using the pixel's to reach position. This cost affects the capability of the algorithm to detect steep skyline curves, especially present in an urban images (formed by buildings).



(a) Vancouver original image

(b) Canny (64,26))

(c) Canny (292,205)

(d) upper envelope

(e) upper envelope

(f) natural skyline

(g) urban skyline

Fig. 1: detailed approach

In [9], an edge detection using Canny filter is applied with high and low scales. Using the topological information, a seed selection step is performed on Canny images allowing to detect the break point at the top of mountains. This seed is detected by searching the maximum point (highest y position) and two local minimum points on its left and right side. The joint angle between the two lines, formed by maximum point and the two loal minimums, validate the seed point. This assumption, that a break point is present, may be verified in some

mountainous areas, but not always in suburb scenario. Finally, the choice of a threshold for joint angle affects the seed point selection and prohibit steep curves to be taken into consideration.

In [10], catadioptric infra-red images are used to detect the skyline for UAV navigation. This method relies on infra-red images which adds robustness to brightness conditions and overcome the sensibility of catadioptric images to luminosity but are not always available, especially in outdoor environments or for a general public application.

In [11], [12] [13], authors situate the work in the context of geolocation in urban canyon, and tested their approach in Fujisawa City. The skyline extraction for geo-location purpose is very useful especially in urban canyons, where high-rise buildings cause satellite blockages, GPS measurement are greatly affected. In this method, upward omni-directional cameras coupled with figure cameras deliver omni-directional images. Sky pixels then become situated in the middle part of the image and not in its uppers. The advantage of using IR-camera is to add robustness to light disturbance.

2.3 Limitations and needs for a new skyline extraction method

We need a skyline extraction algorithm that avoids the main drawbacks of methods presented above: it has to adapt to different depths of interests, making it possible to extract multiple skylines corresponding to near objects or to background objects. It has to handle correctly the steep curves that are very likely to be present in urban landscapes, due to vertical structures of home-made buildings. It has to take into account the fact that the skyline might have several intersections with a single vertical line, in the case of arches, perspective views or overhangs (see non-v-convexity of the skyline, figure 3. Last but not least, its processing time can be a important criterion, as will be explained in our future works, since we plan to use it in real time for a mobile augmented reality application.

3 Proposed algorithm

3.1 Overview

For the rest of the paper, as in the literature, we assume that the skyline is an imaginary curve composed of edge points separating the ground objects and the sky. Added to that, the sky has to be located in the upper part of the image.

Figure 1 shows the steps of our algorithm: we transform the original image to gray scale color space and perform on an edge detection by applying a Canny filter. Manual adjustment of the two Canny filter parameters allows targeting different skylines, according to the subjective definition of each user. Canny edge detection results in many interrupted lines or curves which is followed by a connecting discontinuities step.

In the subsequent sub-sections, we explain detailed procedures.

3.2 Edge detection and upper envelope

In this step, input images are converted to a gray scale color space, then in the domain of the Canny Edge. The user interacts with two sliders to choose its parameters. The choice has to be made so we keep only the most possible pixels that are part of the skyline.

In figure 1b and 1c, depending on the chosen Hysteresis threshold values, we can make two-scale Canny edge images. When choosing low scale values, some pixels (from noisy clutters) which are part of the edge image are taken as skyline's pixels. These edge segment cause later algorithm failure. In figure 2, we give an example where low scales are chosen, and cloud's pixels are taken as skyline's ones.

This parameter choice step allows us to extract what we call an *upper envelope*. In fact, for each column, we keep the highest point in the gradient image: An initial set of points is then obtained. This upper envelope doesn't really define the skyline, but rather a cloud points that, in some cases, reflects the shape of the scene. In this upper envelope, based on human's perception system, we can still recognize the shape of the scene and the skyline due to our ability to link some separated edges pixels. This step of upper envelope extraction is done in real-time and users see it overlapped on the real image.

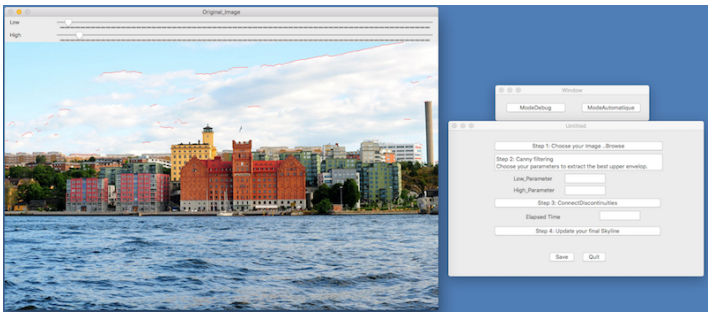


Fig. 2: image with noise

3.3 V-convexity

After upper envelope extraction step, we obtain a set of points that has the disadvantage of being disconnected since two neighboring columns in the image may define points that are vertically very distant (example of steep curve).

A first naive approach would leads us to connect them by segments of straight lines to obtain a continuous curve. This approach is obviously unsatisfactory: In fact, the extracted skyline does not always reflect the real shape of the objects in the image as illustrated in figure 3a. To formalize this problem, we introduced the concept of ***v-convexity***.

A skyline that defines the two region *sky* and *non-sky*, is said v-convex if and only if the intersection of any vertical line with the region *non-sky* is limited to a single line segment.

(a) v-convex approximation (b) non v-convex skyline

Fig. 3: Pearl TV Tower in Shanghai

These cases where the skyline is non v-convex is due to perspective problems, where image present distortion, or image orientation where user take image with non-zero tilt angle. Finally, images can contain artificial objects that are architecturally original (figure 3). The skyline is then called non v-convex. This is also applicable for natural objects, where our algorithm follows the concavities and convexities described by trees, as illustrated in figure 4a and 4b.

(a) result1 (b) result2

Fig. 4: results

We explain in next section (3.4) the proposes method where the previously taken highest points are linked using a shortest path algorithm based on a well-known dynamic programming method, namely *Dijkstra*.

3.4 Connecting Discontinuities

In figure 4a or 4b, green points represents the upper envelope pixels, which have to be connected to obtain a continuous curve defining the skyline. Red line segments represent the followed path after connecting upper envelope points. Figure 3b illustrates an example of our results for the *Oriental Pearl TV Tower* in Shanghai.

Actually, we treat each discontinuity as a weighted graph. First, we use the binary edge map obtained from upper envelope extraction and convert it to a map. In this map, we denote :
w = map's width, h=map's height and each pixel is noted "$p_{i,j}$" where $p_{i,j} = 1$ or 0, with i=∈[1..w] and j=∈[1..h]. Wee then construct a graph G = {V, E, ψ}, where its components are:

1. **V = Vertex:** Each point p_i, j in the map (green pixels in figure 4a or 4b) corresponds to one vertex v_i, j in V ;

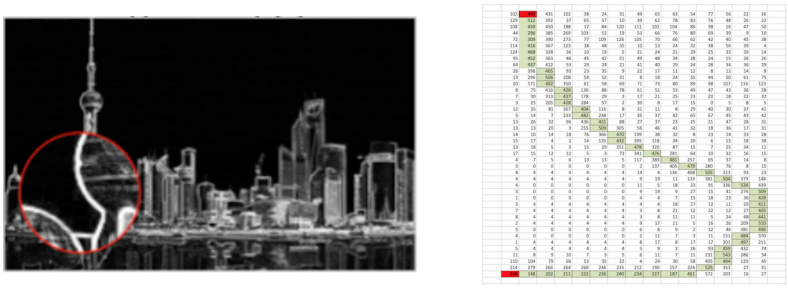
$$V(i,j) = \begin{cases} \textit{Edge vertex}, & \textit{if b(i,j)=1} \\ \textit{Non edge vertex}, & \textit{if b(i,j)=0} \end{cases} \quad (1)$$

2. **E = Edge points:** At adjacent stages, (i) and (i+1), having the two vertices V(i,j) and V(i+1,k), a link has to be established. Thus, depending on a certain criteria, defined below, two cases are possible:
   $|k - j| < 2$: The vertical gap between the two edge vertices is lower than two pixels. In this case, no costs are calculated but a simple line between them is drawn.
   $|k - j| \geq 2$: The vertical gap is larger than two pixels. We use a shortest path-searching algorithm using a *cost function* detailed below, allowing to find a path between the front and the rear end of this sub-graph.
3. ψ **= Cost function** : Costs are calculated using the gradient's magnitude from the filtered original image with *Sobel* filter, as illustrated in figure 5a. In fact, each pixel is connected to its 8 closest neighbors. For each one, we associate a cost value depending on its level's contrast: Lower cost to those with high contrast. A graph is then obtained, allowing to find this shortest path. Using an Excel sheet, we give an example (figure 5b) where red cells are source and destination pixels and green ones correspond to the followed path. In this case, we have a path with 52 pixels size, a source image pixel's coordinate at (184,457) and destination at (185,417).

3.5 Interacting with the given skyline

In some cases, the algorithm fails due to the presence of heavy grey clouds in the image. in fact, some isolated points that have a strong gradient are taken as part of the upper envelope. Thus, the shortest path algorithm tries to link them to next and previous skyline's pixels.

For this, inspired by the *magnetic lasso* in Photoshop or path's modification in *Google Maps* (adding a checkpoint to a path), we give users the ability to



(a) Sobel filtering (b) excel Sheet Path

Fig. 5: cost function

interact with our Graphical User Interface and change one pixel's ordinate in the proposed skyline (up or down). At the same time, while moving the cursor on the interface, the algorithm tries to calculate *two new sub-paths* using the same dynamic programming function: the first, from user's new chosen point previous's area to this last. The second from it (new chosen point) to its next's area. The more and more ordinate amplitude changes increases, neighbouring area expands.

Mobile prototype: Our algorithm runs on desktop in real time. We used and adapted our codes and developed a mobile prototype for extracting the skyline on the move. This prototype allows users to extract the skyline in live video stream while strolling through the city.

4 Results

(a) original image (b) ground truth (c) new ground truth

Fig. 6: CH1 ground truth

In our experiments, different databases were used. Firstly, two databases (namely CH1 and CH2) from [7] containing especially natural images taken in the Alpes. Ground truth images are given only for the first one (CH1). However,

for some of them, we found that the segmentation doesn't really correspond to the real shape of the scene. We give an example in figure 6 illustrating the difference between our new ground truth image and the original one. In fact, the ground truth doesn't take into consideration the concavities and convexities of image elements.

Secondly, we used database from [16], where original and ground truth images are available. Result of the proposed algorithm is also furnished. These images are especially taken in urban areas in the city of *Barcelona*. We could extract skylines accurately in this dataset and compared our result to [16].

All of these images (3 databases 320 images) are taken in various conditions (weather, luminosity, noise..). Only some of them are show in figure 4.

4.1 Assessment of the extracted skyline

First, we define a *discontinuity indice*, giving the distance of the extracted skyline to its ground truth using formula below:

$$\texttt{dist} = \frac{1}{w}\sum_{i=0}^{w} |Sg(k) - Se(k)|$$

where : w = image's width, Sg is the ground-truth Skyline and Se is the extracted one.

We then calculate, for each dataset, the average and the maximum indice. An overview of our results is given in the table 1, giving number of images of each data set, average and maximum CPU-time calculation, average and minimum of Precision and Recall and finally Hamming distance. We compare our method to result of [16] in table 1.

		CH1 [*our*]	CH2 [*our*]	Barcelona [16]	Barcelona [*our*]
images		175	80	52	52
Processing time	average	0,98	0,7 sec		0,71
	max	19 sec	10 sec		5 sec
Precision	average	0,999	0,988	0,959	0,977
	min	0,97	0,293	0,32	0,36
Recall	average	0,99	0,989	0,95	0,55
	min	0,94	0,56	0,95	0,67
Hamming distance	average	2,54	12,59	25,75	23,85
	min	8,31	0,59	143	195

Table 1: results

We conducted our experiments on a desktop machine. Step of extracting the upper envelop is done in real time. Depending on the complexity of the scene, the CPU time needed for the connecting discontinuities step goes from 0,4 to

19 seconds. We also conducted our experiments on a smartphone. Algorithm runs in real-time and allows extracting the upper envelop without latency. As for desktop GUI, connecting discontinuities take more time, and depends also of the smartphone's performance. The developed prototype is ready to be used in an Augmented Reality context.

5 Conclusion and future work

Developed algorithms allow us to extract skyline in real-time on mobile device. This one, called *real skyline* will be compared to a *theoretical* one generated from a 3D City's model. For augmented reality purposes, this comparison will help us correcting the user's pose to more accurate insertion of 3D objects in real-time video stream. We used Lyon's 3D model and are able to extract the theoretical skyline corresponding to user's GPS position. Similar works were done, as in [14], where the image gives pose estimation with panorama's skyline matched with real one, or in [15] where only embedded instruments (GPS, Gyroscope and magnetic compass) were used for the same purpose.

In this paper we have proposed a new interactive system for practical skyline extraction from real images. First, we integrated edge based image segmentation algorithm based on parametric low level image processing. This approach is robust to noise and weather variation. Then, we integrated our approach in a GUI to compare our algorithm's results with user's ones. Finally, we demonstrated that our approach is feasible and adapted it for mobile uses, especially on smartphone, with the intention to use it later for Augmented Reality purpose. Further work are in progress integrating the skyline as a maker for an AR system, which will be later compared with other methodologies and existing approaches as [8] or [15].

Acknowledgments This work was part of the "ANR-12-VBDU-0008 - Skyline" project, funded by the "Agence Nationale de la Recherche (ANR)" and the Labex (Laboratoire d'Excellence) "Intelligence des mondes Urbains (IMU)".

References

1. Johns, D. and Dudek, G.: Urban Position Estimation from One Dimensional Visual Cues. In: 3rd Canadian Conference on Computer and Robot Vision (CRV'06), pp. 22–22, IEEE (2006)
2. Yusoff, N.A.H. and Noor, A.M. and Ghazali, R.: City Skyline Conservation: Sustaining the Premier Image of Kuala Lumpur. In: Procedia Environmental Sciences, pp.83–592, Elsevier B.V (2014)
3. Fang, M. and Chiu, M.-Y. and Liang, C.-C. and Singh, A.: Skyline For Video-based Virtual Rail For Vehicle Navigation. In: Proceedings of the Intelligent Vehicles '93 Symposium, pp.207–212, IEEE (1993)

4. Byung-Ju, K. and Jong-Jin, S. and Hwa-Jin, N. and Jin-Soo, K.: Skyline Extraction using a Multistage Edge Filtering. In: International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering. Vol.5, 10 - 14 (2011)
5. Jiebo, L and Etz, S.P.: A physical model-based approach to detecting sky in photographic images. In: IEEE Transactions on Image Processing, Vol.11, 201–212 (2002)
6. Saurer, O. and Baatz, G. and Köser, K. and Ladický, U. and Pollefeys, M: Image Based Geo-localization in the Alps. In: International Journal of Computer Vision. Vol.116, 213–225 (2016)
7. Baatz, G. and Saurer, O. and Koser, K. and Pollefeys, M. : Large Scale Visual Geo-Localization of Images in Mountainous Terrain. In: European Conference on Computer Vision, pp.517–530, (2012)
8. Lie, W. and Lin, T.C.-I. and Lin, T. and Hung, K-S.: A robust dynamic programming algorithm to extract skyline in images for navigation. Pattern Recognition Letters. Vol.26, 221–230 (2005)
9. Yang, S.W. and Kim, I.C. and Kim, J.S.: Robust skyline extraction algorithm for mountanious images. In: Proceedings of the Second International Conference on Computer Vision Theory and Applications,pp.253–257, SciTePress - Science and and Technology Publications (2007)
10. Bazin, J.-C. and Kweon, I. and Demonceaux, C. and Vasseur, P: Dynamic programming and skyline extraction in catadioptric infrared images. In: IEEE International Conference on Robotics and Automation, pp. 409–416, IEEE, (2009)
11. Meguro, H-I. and Murata, T. and Amano, Y. and Hasizume, T. and Takiguchi, J-I.: Development of a Positioning Technique for an Urban Area Using Omnidirectional Infrared Camera and Aerial Survey Data. In: Advanced Robotics, pp. 731–747, (2008)
12. Ramalingam, S. and Bouaziz, S. and Sturm, P. and Brand, M.: SKYLINE2GPS: Localization in urban canyons using omni-skylines. In International Conference on Intelligent Robots and Systems, pp. 3816–3823, IEEE (2010)
13. Ramalingam, S. and Bouaziz, S. and Sturm, P. and Brand, M.: Geolocalization using skylines from omni-images. In: the 12th International Conference on Computer Vision Workshops, ICCV Workshops, pp. 23–30, IEEE (2009)
14. Zhu, S. and Morin, L. and Pressigout, M. and Moreau, G. and Servieres, M.: Video/GIS registration system based on skyline matching method. In: International Conference on Image Processing, pp.3632–3636, IEEE, (2013)
15. Fukuda, T. and Zhang, T. and Yabuki, N.: Improvement of registration accuracy of a handheld augmented reality system for urban landscape simulation. Frontiers of Architectural Research. Vol.3, 386–397 (2014)
16. Tighe, J. and Lazebnik, S. : SuperParsing: Scalable Nonparametric Image Parsing with Superpixels. In: European Conference on Computer Vision, pp 352–365 (2010)