Adaptive Rendering for Large-Scale Skyline Characterization and Matching

Jiejie Zhu, Mayank Bansal, Nick Vander Valk, and Hui Cheng

Vision Technologies Lab., SRI International, Princeton, NJ 08540, USA
{jiejie.zhu,mayank.bansal,nicholas.vandervalk,hui.cheng}@sri.com

Abstract. We propose an adaptive rendering approach for large-scale skyline characterization and matching with applications to automated geo-tagging of photos and images. Given an image, our system automatically extracts the skyline and then matches it to a database of reference skylines extracted from rendered images using digital elevation data (DEM). The sampling density of these rendering locations determines both the accuracy and the speed of skyline matching. The proposed approach successfully combines global planning and local greedy search strategies to select new rendering locations incrementally. We report quantitative and qualitative results from synthesized and real experiments, where we achieve a computational speedup of around 4X.

1 Introduction

Skylines, especially in mountainous areas, provide robust and often unique features to characterize an area. Often by looking at the skyline in a photo, such as the half dome in Yosemite, one can recognize the area where the photo was taken. However, to recognize where a photo is taken over a large area of tens or hundreds thousands of square kilometers is very difficult. Based on this observation and previous work [1][2] on skyline analysis, we developed a large-scale skyline characterization and matching system for automated geo-tagging of photos and images.

Given an image, our system automatically extracts the skyline and then matches it to a database of reference skylines extracted from digital elevation data (DEM). The reference skylines are generated through rendering using DEM at locations over the entire area of interest. The selection of these rendering locations determines both the accuracy and the speed of the skyline matching. In addition, it determines the time needed to build the reference skyline database.

We can use a grid on the ground-plane as rendering locations – in this case the distance between two adjacent grid locations is the sampling distance. The larger the sampling distance is, the faster the rendering and the matching algorithms run, but less accurate the skyline matching is. The smaller the sampling distance is, the more accurate the matching is, but the slower the rendering and the matching algorithms run. For example, skyline rendering and extraction over a 10,000 km^2 using 50-100 meters sampling distance can take months to complete. On the other hand, if the sampling distance is large, photos taken close to a mountain or inside a mountain, such as those taken on a hiking trail or a road

A. Fusiello et al. (Eds.): ECCV 2012 Ws/Demos, Part I, LNCS 7583, pp. 163–174, 2012.

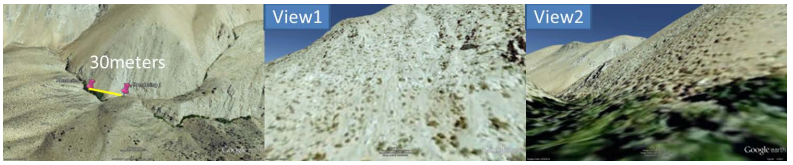

Fig. 1. Dramatic skyline change caused by occlusions in complex mountainous terrain

passing through a mountain cannot be found. Fig. 1 depicts such an example where two views that are separated by only 30 meters have very different skylines.

In this paper, we propose an adaptive rendering approach for optimal rendering location selection. By modeling both the rendering process and the skyline matching process, our system can compute the set of optimal rendering locations based on the DEM and a pre-defined matching threshold. The viewpoint selection is optimal in the sense that for a given matching accuracy, it requires far fewer number of renderings than if the rendering locations were uniformly sampled on a dense grid. In the example of Fig. 1 above, a good viewpoint selection algorithm should automatically render densely when close to the mountain and coarsely when far away.

2 Related Work

In recent literature, there has been increasing interest in geo-localization of ground-level imagery using visible skylines by matching them to a database of known skyline shapes. For urban geo-localization, Srikumar et al.[3][2] focused on matching omni-skylines from an upward facing camera to skyline renderings generated on-the-fly from 3D building models of the scene. Our focus in this paper is on geo-localization in natural terrain – in this case, the problem is much harder since we have to match to a specific viewpoint (i.e. the query camera is not upward facing) and we cannot pre-render on sparse road networks like in the urban case. It therefore becomes important to devise a rendering scheme that will render at the fewest number of viewpoints without reducing the localization accuracy achievable.

Lionel et al.[1] recently addressed the problem of automatic photo-to-terrain alignment with a goal of annotating mountain pictures. In their work, high resolution elevation maps are rendered to create synthesized panoramic views which are then matched to the mountain picture using a robust edge matching algorithm. However, their approach assumes that the GPS location and FOV of the query picture are known and then they solve for the unknown camera pose relative to the terrain. In contrast, we would like to address the problem of localization of the query picture itself by matching to a set of terrain renderings obtained from sampled viewpoints. To achieve this, we have to either sample these viewpoints very densely everywhere or we can adopt an adaptive rendering strategy to minimize the search required at the matching stage. In this paper, we focus on such an approach.

Bryan et al.[4] address the problem of automatically aligning historical architectural paintings with 3D models obtained from modern photographs. A key step in their approach is the "view-sensitive retrieval" that aims to find a 3D viewpoint that is sufficiently close to the painting viewpoint. To achieve this, they sample a large (dense) pool of virtual viewpoints around the 3D model and then use a matching procedure to retrieve a small set of nearby matching candidate viewpoints. In this paper, we propose an approach to allow a non-dense rendering of the 3D scene for query localization in the context of natural terrain where dense sampling quickly becomes a computational bottleneck.

There has been some work in the literature on efficient rendering of large terrains. However, the majority of this work has focused on the graphics aspect of rendering including adaptive means to enable faster Google-Earth like renderings served to an end-user. For example, Raphael el al.[5,6] describe a generic data structure to adaptively serve data to the client rendering system and to improve the database loading and rendering speeds independent of the database size. Similarly, a hardware accelerated terrain rendering approach is outlined in [7]. In this paper, we assume that a terrain rendering algorithm is available to us as a black-box and we can use it to render the terrain at any specified location and viewpoint.

3 Skyline Rendering and Matching

In this section, we briefly introduce our skyline rendering and matching algorithms. In skyline rendering, the system takes an area of geo-localized DEM data as input. A ground-level camera location inside this area is specified along with its intrinsic and extrinsic parameters. The system then renders the DEM into a depth image, which is then used to extract a skyline corresponding to the specified viewpoint e.g. Fig. 2.

Given all the rendered skylines, the system extracts features from each of them and saves them in a database for future matching purposes. In feature extraction, a skyline is first approximated by polylines and then the end-points of the line-segments composing these polylines are used as feature points.

Given a query image like in Fig. 3, we use a skyline extraction algorithm [2] to extract the skyline. The polyline approximation-based algorithm described above is then used to extract the query skyline features. For each extracted key feature from the query skyline, the matching process finds a key feature from the rendered skyline that best matches it. The matching score between a keypoint pair is computed as the Chamfer distance between the local skylines centered at these keypoints. This establishes a correspondence between the keypoints of the two skylines following which RANSAC is used to find inliers corresponding to an affine transformation between the two skylines. The overall matching score between the two skylines is computed as the Euclidean distance between the skylines after warping with the computed affine transformation.

Fig. 2. Example of a rendered depth image and its extracted skyline. To extract the skyline, the system looks at the depth image from top to bottom in each column, and labels as skyline pixel the first pixel with a non-infinite depth.

Fig. 3. Example of a query image and its skyline. The contour of the trees are false skyline pixels automatically computed from the skyline extraction model – the RANSAC-based matching process labels them correctly as outliers.

4 Problem Formulation

We assume that we start the adaptive rendering algorithm from a set of pre-rendered viewpoints which are on a uniform but coarsely sampled grid, e.g. at 1 Km spacing. The goal of adaptive rendering is to automatically predict optimal viewpoints c_k (with spacing finer than 1 Km) at which the rendered skyline looks sufficiently different from existing renderings of the same 3D feature (mountain). To achieve this goal, we approach the viewpoint selection process as an incremental algorithm that adds new viewpoints to the set of existing renderings, *but without rendering the new viewpoints first.* Around each already rendered skyline from a viewpoint c_0, we can explicitly compute a "tolerance area" within which the skyline projection from another candidate viewpoint c_k looks similar to c_0. This similarity is defined by an image projection distance between the skylines at c_k and c_0. Thus, for a given threshold on the similarity metric, we can estimate the tolerance area around each existing rendering. Intuitively, the tolerance area specifies the extent within which no additional rendering is required. Thus, for an existing viewpoint inside a complex terrain, we can expect to obtain a much smaller tolerance area than for a simple flat terrain. The tolerance area computation algorithm is described in Sec. 4.2. Given the tolerance areas (and their shape) for each of the existing renderings in our set, in Sec. 4.3, we propose and compare several novel planning strategies to determine the next-best viewpoint that avoids overlap with the existing tolerance areas.

4.1 Camera Configuration

We assume a distortion-free ideal pin-hole camera model for the rendering camera with square pixels and a camera center coincident with the image center. This leads to an ideal camera intrinsic matrix. For our experiments, we cover the 360 degree view surrounding each candidate rendering location using four camera viewpoints $v_1, \ldots, v_4$, each with a horizontal field-of-view of $\theta = 90$ degrees and image resolution of $w = h = 640$. The optical axis for each viewpoint v_i can be described by a pre-defined rotation matrix R_i which fixes a single look-at direction for this viewpoint independent of the rendering location.

In our framework, we represent the 3D points corresponding to the skylines (mountain silhouettes) in world-coordinates by variable $X_i \in \Re^3$. The skyline projection at any viewpoint c_k is then given by:

$$I_k = P_k X_i = K[R|t_k]X_i$$

where $P_k = K[R|t_k]$ is the camera projection matrix for the camera located at displacement t_k w.r.t the world coordinate system origin. The rotation R is one of $R_1, \ldots, R_4$ depending on the viewpoint's look-at direction.

In the following, we will follow the convention that the world coordinate system is defined with XZ as the ground-plane and the Y-axis pointing upwards.

4.2 Tolerance Area Computation

The squared difference between projected skylines I_0 and I_k (at locations c_0 and c_k respectively) can serve as a simple error metric for the estimation of tolerance area. However, the skyline matching algorithm (Sec. 3) accounts for any small distortions of the skyline by an affine transformation model. It therefore makes sense to measure the projection error between skylines visible at two viewpoints after allowing for an affine transformation. Fig. 4 illustrates an example of two projections of a synthetic skyline where the distance between the original skylines does not correctly reflect the difference between their shape. Warping one of the skylines using an affine transformation, however, leads to a much more accurate error metric.

Estimating affine transformation between two images has been studied extensively. Most techniques rely on first applying RANSAC [8] to determine the corresponding pixels and then to recover their geometrical relationships by rejecting outliers. Here, we can apply a similar approach but the computation cost is high considering the huge number of such pair comparisons required in the system.

We propose an analytical approximation approach to compute the affine transformation between the two cameras given their projection matrix. It has merits of requiring much less computation (since it is analytical) while preserving the accuracy compared with the traditional correspondence-based estimation method.

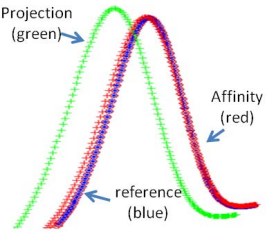

Fig. 4. Example of skyline projection improvement using an affine transformation model from a synthetic experiment. The green and blue curves are two candidate skylines; the red curve is the green curve after warping by an affine transformation that best aligns the candidate skylines. The direct Euclidean distance between the green and blue skylines is around 26 pixels. The corresponding distance between the red and blue curves is 3.6, which is a more accurate estimate of the difference between the skyline shapes than the direct distance.

Affine Transformation. Given two projection matrices $P_1 = K_1[R_1|T_1]$ and $P_2 = K_2[R_2|T_2]$, without loss of generality, we can assume the first camera to be in a canonical form such that $R_1 = I, T_1 = 0$. Instead of selecting all four viewing directions $v_1, \ldots, v_4$ simultaneously for each location, we apply the adaptive rendering selection algorithm to viewpoints facing each direction individually – implying $R_2 = R_1$. Further, for our problem, $K_2 = K_1 = K$. If a 3D point X is visible to both cameras, its projection can be expressed by $U_1 = KX$ and $U_2 = KX + KT$, where $T = T_2 - T_1$. By substituting X using $K^{-1}U_1$, we have

$$U_2 = U_1 + KT \tag{1}$$

By representing U_1 in homogeneous coordinates U_1', $K = [k_1, k_2, k_3]'$ and writing the above equation in a matrix formulation, we have

$$U_2 = \begin{pmatrix} k_3X & 0 & k_1T \\ 0 & k_3X & k_2T \\ 0 & 0 & k_3X + k_3T \end{pmatrix} \begin{pmatrix} \frac{k_1X}{k_3X} \\ \frac{k_2X}{k_3X} \\ 1 \end{pmatrix} \tag{2}$$

If we represent U_2 in homogeneous coordinates U_2', we have

$$U_2' = \frac{U_2}{k_3X + k_3T} = AU_1' \tag{3}$$

$$= \begin{pmatrix} \frac{k_3X}{k_3X+k_3T} & 0 & \frac{k_1T}{k_3X+k_3T} \\ 0 & \frac{k_3X}{k_3X+k_3T} & \frac{k_2T}{k_3X+k_3T} \\ 0 & 0 & 1 \end{pmatrix} U_1' \tag{4}$$

In the above equation, we should note that k_3X is the displacement between the 3D point X and the reference (P_1) camera center, k_3T is the displacement between the 3D point X and the candidate (P_2) camera center. k_3X will be different for each 3D point located on the skyline if their Z-coordinates are

different. Building a per-point affine transformation generates the exact transformation between U_1 and U_2, but it is computationally expensive given the large number of 3D points on skylines. Instead, our system chooses a z-value that is representative of the skyline such as the mean or median distance of the skyline points.

Given the above formulation, we now outline our tolerance area computation algorithm: We sample a 10×10 neighborhood around the reference location c_0. At each sampled location, we compute the reprojection error of the skylines between the reference and the sampled location using Eq.4 – if this error is within a given threshold, the sampled location is included in the tolerance area (corresponding to the reference location c_0), otherwise it is excluded.

4.3 Optimal Viewpoint Planning and Selection

Without loss of generality, we assume that we have already computed the tolerance areas for the four corners of a 1 Km $\times$ 1 Km square on the terrain using the algorithm in the previous section. The next step in our pipeline is to select a new location at which the renderer should render so as to cover maximum uncovered ground. In the following, we describe five different strategies to plan the next viewpoint. In the experiments section, we will discuss how each of these strategies performed on our simulation data followed by results of the best performing strategy on real data.

Random Optimistic Viewpoint (ROV). ROV randomly chooses a number of candidate viewpoint locations from the uncovered area. Their tolerance area are computed and ranked based on their overlap with the uncovered area. ROV selects the location with the highest coverage and performs rendering at the selected location. This process will iterate until the coverage reaches an acceptable number, e.g. 90% or 98% used in the experiments.

Approximated Random Optimistic Viewpoint (AROV). The tolerance area computation in ROV is expensive since it requires projecting each 3D skyline point at multiple candidate locations (please refer to the computation cost in the last row of Table 1). To make this process more efficient, AROV interpolates a candidate viewpoint's tolerance area linearly from its nearest rendered viewpoints whose tolerance areas are already computed.

Hierarchy Optimistic Viewpoint (HOV). Different from ROV and AROV, HOV searches optimal viewpoint using a coarse to fine process. At each level, AROV is used to select the next-best viewpoint; when coverage is satisfied at a level, HOV will move to the next finer level.

Shape-Assisted Optimistic Viewpoint (SOV). SOV investigates how the shape of the tolerance area may be used in assisting optimal viewpoint selection. From ROV, AROV and HOV, we noticed that most of tolerance areas can be best described using 2D ellipses. Thus, instead of randomly picking the candidate locations, in SOV we pack them along the minor axes of the ellipse-like tolerance areas so as to achieve a tighter packing over the uncovered area.

Line Planning Optimistic Viewpoint (LPOV). SOV uses independent tolerance areas to select the optimal viewpoint and this strategy may not be optimal for covering the whole area. In order to include more global information in selecting an optimal viewpoint, LPOV pre-locates optimal viewpoints on each line joining a pair of sampled viewpoints. The location of a candidate viewpoint is predicted as a convex combination of the locations of the endpoints; the weights in this convex combination are proportional to the size of each end viewpoint's tolerance area. LPOV encourages sparse optimal viewpoint selection in flat areas since the tolerance areas of the rendered locations are larger in comparison with the rendered viewpoints in clutter areas, such as inside mountains. This line-wise planning and local greedy combined approach contributes to less number of rendered viewpoints with a fast coverage.

5 Experimental Results

This section exhibits our quantitative and qualitative experimental results on i) a synthetic dataset generated by sampling skylines from a parametric model, and ii) a DEM dataset covering around 50 km^2 on a mountainous terrain.

5.1 Simulation Results

We evaluated the five viewpoint selection techniques described in Sec. 4.3 on a synthetic dataset consisting of an area with 100×100 potential (dense) viewpoint locations. All the experiments start with four rendered viewpoints at the corners initially. We evaluate the performance of the algorithms using two criteria:

(A) Total number of viewpoints selected for rendering and,
(B) Size of overlapped area – computed as the number of times the same viewpoint location is included in any of the tolerance areas.

The goal of a good adaptive rendering algorithm is to cover as much area as possible using minimal number of rendered viewpoints where the coverage is defined as the union of tolerance areas from all rendered viewpoints. Thus, our objective is to achieve smaller numbers for both criteria (A) and (B).

Table-1 reports the results of all the methods for the 100×100 grid using the convention x/y where x represents the value of criterion (A) and y the value of criterion (B). We include results from experiments with skylines of different complexity with the reprojection error evaluated with and without the affine transformation model proposed in this paper.

We can conclude the following from the simulation results:

1. Overall, the affine transformation model requires fewer number of renderings, but produces more overlaps, as shown by a comparison between row 1 and rows 2,3.
2. Overall, the number of selected viewpoints is proportional to the complexity of the skylines as observed by comparing row 2 and row 3.

Table 1. Simulation results for viewpoint planning methods using criteria (A)/(B)

Parameters	ROV	AROV	HOV	SOV	LPOV
Non-Affine (p=1, t=15, c=90% r=0)	**240**/1545	286/1636	324/**1210**	275/1941	325/1515
Affine (p=1, t=3, c=98% r=0)	81/6664	**80**/5824	98/6723	100/10043	86/**5489**
Affine (p=2, t=3, c=98%, r=30)	239/9028	217/6991	252/7226	245/9148	**191/5943**
Computation Cost (rounded)	45mins	10mins	5mins	30mins	10mins

p: number of peaks on the skyline; t: reprojection error threshold in pixels
r: skyline rotation angle; r=0 represents a skyline that is parallel to the camera plane.
c: coverage percentage at which the adaptive rendering process is terminated.

Table 2. Query geo-localization accuracy results with and without adaptive rendering

	Rendering Resolution/Number of renderings				
	1024m/182	512m/665	256m/2662	128m/10648	Adaptive/2800
Close to mountain	1906m	800m	379m	64m	127m
Medium Range	1609m	703m	402m	102m	98m
Far from mountain	1247m	604m	453m	202m	125m

3. SOV consistently produces the largest size of overlaps which shows that locally optimal algorithms do not achieve good results.
4. ROV requires the highest computation cost. Other algorithms require less computational since the tolerance area is approximated using nearest neighbors. Among them, HOV requires the least computation because a fixed pattern of optimal viewpoint selection is used.
5. LPOV obtains the best performance in the test using skylines with two peaks. This suggests that a planning strategy with a local greedy objective may give good results.

5.2 Experiments Using DEM Data

For experiments with real data, we used an area approximately 5km×10km in size in a moutainous terrain. We select the best performing viewpoint selection method from the simulation (LPOV) for experiments with real data and compare its performance with a hierarchical uniform approach. To characterize the performance of the generated renderings using either approach, we use a set of query images (with known ground-truth geo-location information) shown in Fig. 5 and match each of them to the generated renderings. We selected three groups of query images: close to the mountain (Fig. 5, column-1), medium range from the mountain (Fig. 5, column-2) and far from the mountain (Fig. 5, column-3). We measure the distance between the location of the best matching rendering and the known ground-truth location of the query to showcase the improvement in geo-location using renderings from our adaptive algorithm versus renderings on a uniform grid at four different levels of resolution (1024m, 512m, 256m and 128m).

Fig. 6 shows an example of the viewpoints selected from the proposed adaptive rendering approach with two headings at 0 and 90 degrees (similar results are

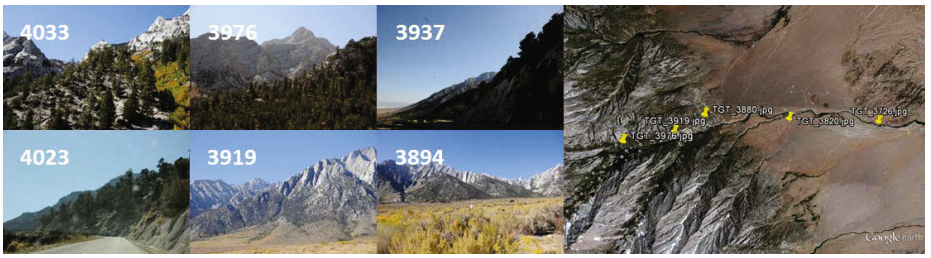

Fig. 5. 6 query images used in all the experiments, along with their geo-locations on the overhead view

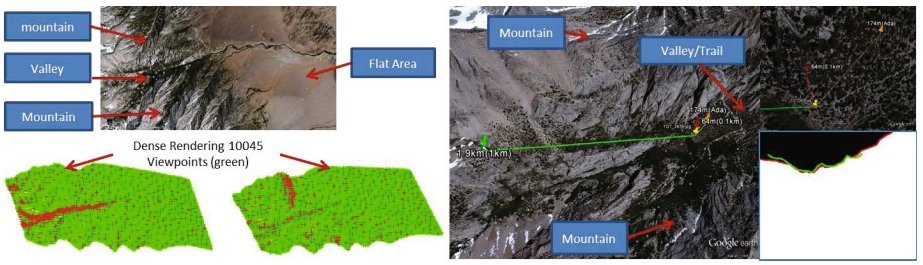

Fig. 6. Left: Example of viewpoints selected from the proposed adaptive rendering approach for heading 0 and 90 degrees. Looking at the terrain area shown on the top, we can see that a large number of viewpoints are selected in the valleys and areas with clutter while a fewer number of viewpoints are selected by our algorithm in flat areas of the terrain. It is also interesting to see that different heading directions will result in different adaptive rendering results given the terrain structure. Right: Result of query 3976. In uniform viewpoint sampling, a 1024m spacing generates best geo-location around 1900m away, while a 128m spacing gives the best geo-localization around 64m away. Adaptive rendering has an error of 174m, but with a 400% system performance improvement. The inlay on the bottom right illustrates the skyline matching result for this query. The green curve indicates the skyline in the query image. The red curve shows the skyline in the rendered image.

obtained from the other two headings). We can see that dense viewpoint sampling is required in cluttered terrain areas such as valley and trails. Incidentally, these areas are critical to skyline matching because occlusions introduce a large number of mismatches.

Skyline matching tests are performed for query images shown in Fig. 5. One of the geo-localization results is shown in Fig. 6. The numerical results are reported in Table 2.

Fig. 7 highlights the important fact that using the proposed adaptive rendering algorith, one can achieve a given matching accuracy using only a small fraction of the number of renderings required by a uniform fixed renderer.

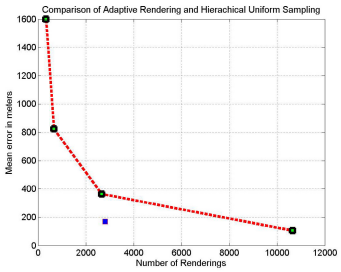

Fig. 7. Comparison of skyline matching accuracy using adaptive rendering and uniform viewpoint sampling at multiple resolution levels. In terms of accuracy, adaptive rendering leads to an improvement of almost 200%. It has a mean error of 200m with the number of renderings around 2500 while the error increases to 400m with the same number of renderings in the case uniform sampling. In terms of the system performance, adaptive rendering achieves 300% improvement as uniform sampling requires more than 7000 renderings to bring the mean error down to 200m.

6 Conclusion

We have proposed an adaptive rendering based approach to enhance geo-localization from 2D skyline images. Using affinity error metric, viewpoint optimality and line-wise planning presented in this paper, our method can successfully reduce overall 4X computational cost while preserving the geo-localization accuracy compared with a uniform sampling approach. In the near future, we would like to explore testing on a larger area. We expect to see even further improvements since terrain surfaces in large areas will likely contain more scattered complex areas where a uniform rendering solution will be infeasible for a desired level of geo-localization accuracy.

References

1. Baboud, L., Cadik, M., Eisemann, E., Seidel, H.: Automatic photo-to-terrain alignment for the annotation of mountain pictures. In: 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 41–48. IEEE (2011)
2. Ramalingam, S., Bouaziz, S., Sturm, P., Brand, M.: Skyline2gps: Localization in urban canyons using omni-skylines. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3816–3823. IEEE (2010)
3. Ramalingam, S., Bouaziz, S., Sturm, P., Brand, M.: Geolocalization using skylines from omni-images. In: 2009 IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops), pp. 23–30. IEEE (2009)
4. Russell, B., Sivic, J., Ponce, J., Dessales, H.: Automatic alignment of paintings and photographs depicting a 3d scene. In: 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), pp. 545–552. IEEE (2011)

5. Lerbour, R., Marvie, J., Gautron, P.: Adaptive streaming and rendering of large terrains: A generic solution. In: Proceedings of WSCG (2009)
6. Lerbour, R., Marvie, J., Gautron, P.: Adaptive real-time rendering of planetary terrains. In: Proceedings of WSCG (2010)
7. Röttger, S., Ertl, T.: Hardware accelerated terrain rendering by adaptive slicing. In: Workshop on Vision, Modelling, and Visualization VMV, vol. 1, pp. 159–168 (2001)
8. Choi, S., Kim, T., Yu, W.: Performance evaluation of ransac family. In: BMVC (2009)