

SKYLINE2GPS: Localization in Urban Canyons using Omni-Skylines

Srikumar Ramalingam^{1*} Sofien Bouaziz^{2*} Peter Sturm³ Matthew Brand¹

¹Mitsubishi Electric Research Lab (MERL), Cambridge, MA, USA

²Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland

³INRIA Grenoble – Rhône-Alpes and Laboratoire Jean Kuntzmann, Grenoble, France

{srikumar.ramalingam,brand}@merl.com, sofien.bouaziz@gmail.com, peter.sturm@inrialpes.fr

Abstract—This paper investigates the problem of geo-localization in GPS challenged urban canyons using only skylines. Our proposed solution takes a sequence of upward facing omnidirectional images and coarse 3D models of cities to compute the geo-trajectory. The camera is oriented upwards to capture images of the immediate skyline, which is generally unique and serves as a fingerprint for a specific location in a city. Our goal is to estimate global position by matching skylines extracted from omni-directional images to skyline segments from coarse 3D city models. Under day-time and clear sky conditions, we propose a sky-segmentation algorithm using graph cuts for estimating the geo-location. In cases where the skyline gets affected by partial fog, night-time and occlusions from trees, we propose a shortest path algorithm that computes the location without prior sky detection. We show compelling experimental results for hundreds of images taken in New York, Boston and Tokyo under various weather and lighting conditions (daytime, foggy dawn and night-time).

I. INTRODUCTION AND PREVIOUS WORK

The skyline has long been a source of fascination for photographers, thought to identify the city as uniquely as a fingerprint. In this work, a step further has been taken to investigate its uniqueness on omni-images for any given geospatial location. Such an approach has several interesting properties compared to existing techniques. First, the skyline is compact, unique and extremely informative about a location. Second, it is less prone to occlusions (trees, lampposts, cars and pedestrians) and lighting variations than other features. The main reason to use omni-images instead of perspective ones comes from the fact that the stability of the underlying pose estimation problem increases with the field of view of the camera. It has also been proven formally that omni-directional cameras can give much better accuracy in motion estimation than perspective ones [6], [3]. In the case of small rigid motions, two different motions can yield nearly identical motion fields for classical perspective cameras, which is not the case for omnidirectional images.

In robotics and vision community, several promising simultaneous localization and mapping (SLAM) algorithms have been developed in the last three decades and detailed surveys are available [8]. Existing techniques in SLAM can be classified into ones that use a motion model [2], [4] and the approaches free of motion models [20], [26]. The basic idea in using a motion model is to smooth the trajectory of the camera and constrain the search area for

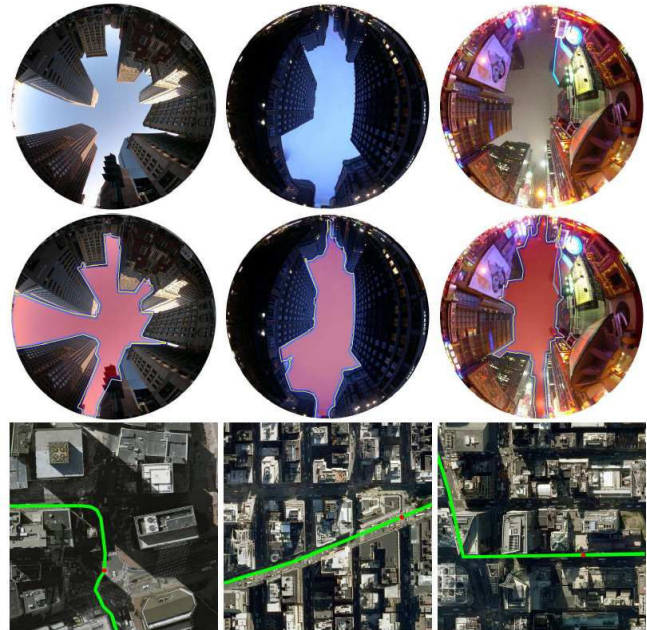


Fig. 1. On the top we show upward facing images of skylines. In the middle, we show the matched skylines using the proposed algorithms. In the bottom, we show the corresponding geo-locations on aerial images. [Best Viewed in Color]

feature correspondences. On the other hand, the ones without using a motion model reconstruct the scene coarsely using 3D reconstruction algorithms and estimate the pose of the camera w.r.t the coarse model. In contrast to many methods where both the 3D reconstruction and localization are solved simultaneously or sequentially, our method attempts to solve only the localization problem assuming that a coarse 3D model of the city is already given.

In the last few years, there has been an increasing interest in inferring geolocation from images without using SLAM-style algorithms [25], [33], [13], [10]. Several recent image based localization approaches have been proposed using feature matching algorithms. The general idea behind most of these approaches is to find the closest image to a given query picture in a database of GPS-tagged images. Robertson and Cipolla [25] showed that it is possible to obtain geospatial localization by wide-baseline matching between a new image from a mobile camera and a database of rectified building facade images. Zhang and Kosecka showed accurate results

*joint primary authors

in the ICCV 2005 computer vision contest (“Where am I?”) using SIFT features to search the closest images in a database of GPS-tagged images [33]. Jacobs et al. [13] used an interesting approach to geolocate a static outdoor webcam by correlating its images with satellite weather imagery taken at the same time of day or with other webcams’ images for which the locations are known. The underlying idea is that there is a consistent pattern in the manner in which these images vary over time. Hays and Efros [10] used millions of GPS-tagged images from the web for georeferencing a new image. In order to match the query image with a large collection of images (6 million images), several features based on color histograms, texton histograms, line features, gist descriptor, geometric context (labeling of the image pixels into sky, buildings and ground), etc. are used. The use of time-stamped photographs has also proved beneficial for geo-localization [14].

In contrast to most of these approaches that leverage on the availability of these georeferenced images, the approach presented uses coarse 3D models downloaded from the web. A large repository of coarse 3D models already exists for major cities. There exist a few more related approaches that use 3D models and/or omni-directional cameras for geolocalization [16], [28], [18], [31], [5]. Koch and Teller proposed a localization method using a known 3D model and a wide angle camera for indoor scenes by matching lines from the 3D model with the lines in images [16]. In contrast to their technique, this work relies on the skylines for geolocalization of outdoor scenes. Although, the idea of using the horizon or the skyline has been explored earlier in [28], [18], [1], our approach is different from them in several ways. In [28] a human user input is required to extract the skyline, whereas this work uses an automatic graph cuts algorithm. In addition, we also propose a method which does not need prior sky detection. In [28] a hash table is precomputed to match the unwrapped horizon with the 3D model, whereas the algorithm presented in this work synthesizes fisheye images on the fly for matching. This relaxes the requirement of pre-processing and storing millions of synthetic skylines. Our work is closest to [18], [1], where an omni-directional infrared camera is used instead of an omnidirectional visible camera or a prior sky detection is necessary for the algorithm to work. This paper is an extended version of our previous workshop paper [21].

II. OVERVIEW OF OUR APPROACH

We briefly explain the various building blocks of our geolocalization system.

Calibration and fisheye synthesis. In this work we used a fisheye lens with a field of view of about 183° . However, we would like to propose a method that could work with all kinds of omni-directional cameras. For that reason, we chose to use a generic calibration approach which treats every camera as a mapping between a pixel and its corresponding projection rays. It was recently shown that the generic calibration algorithm outperforms standard parametric approaches in the case of very high distortions [23], [7]. We

are not aware of any work in vision which synthesizes omni-directional images from 3D models for pose estimation. The synthesis is done using a pixel shader program implemented on a Graphics processing unit (GPU) for generating a large number of fisheye images in real time.

Skyline matching using graph cuts and robust chamfer distance. We propose a graph cuts based algorithm embedded with parameter learning for robustly detecting sky in omni-directional images. The most closely related work, which already gives excellent results, is the geometric labeling problem to classify a given image into sky, buildings and ground [12]. After detecting the sky, the skyline matching is done using the chamfer distance embedded in a RANSAC framework.

Skyline matching using shortest path algorithm. Through a rigorous study we found that the sky detection method proposed will not be robust for all weather conditions. In order to handle difficult conditions due to weather and lighting changes we also propose a *max-min* operator and a shortest path algorithm to compute the location. Here we relax the requirement of sky-detection for geo-localization. On the other hand, we define a cost function that validates the various skyline proposals from the 3D model.

Experiments. We show promising experimental results for hundreds of images captured under various weather and lighting conditions in different cities. We also show that our method is capable of outperforming GPS measurements in urban canyons.

Degeneracy analysis. Finally we show a minimal solution for 3DOF pose estimation using the minimal number of required features (points and lines) on the skylines. This theory highlights the important connections of the 2D-3D registration problem to our proposed skyline-matching and clearly points out the possible degenerate cases and rarity of such cases in practice.

A. Calibration and Fisheye Synthesis

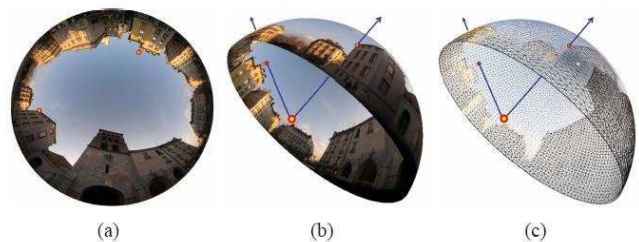


Fig. 2. (a) A fisheye image with two pixels marked. (b) and (c) show the projection rays corresponding to the two pixels from the hemispherical set of rays computed using generic calibration.

We use a generic imaging model to calibrate our fisheye lens [23]. According to this model, every pixel in the omni-image is mapped to a 3D half-ray in space. This mapping can be computed using images of calibration grids. It has been shown recently that the use of generic imaging models allows very precise distortion correction [23].

In Figure 3 we show the various stages in synthesizing a fisheye image. In order to synthesize a fisheye image we first generate five binary perspective images, corresponding to the views in the hemicube shown in Figure 3(c). As our algorithm uses coarse 3D models without any texture, we generate a cubemap simply by rendering a 3D model colored entirely in white; the resulting binary image is black in sky regions. We then use our calibrated ray-table to map the cubemap to a fisheye image, as shown in Figure 3(c). The black region is the predicted shape of the sky. Some of the fisheye images synthesized at different places in a 3D model of Boston’s financial center are shown in Figure 3(d).

As the previous section indicated, our ray-calibrated view synthesis has the advantage that it does not “bake in” errors that would arise from using a parametric lens model. In addition, a pixel shader program is implemented in GPU to generate these fisheye images at a very fast rate. This allows us to generate accurate fisheye images on the fly; there is no need to store images in a large database.

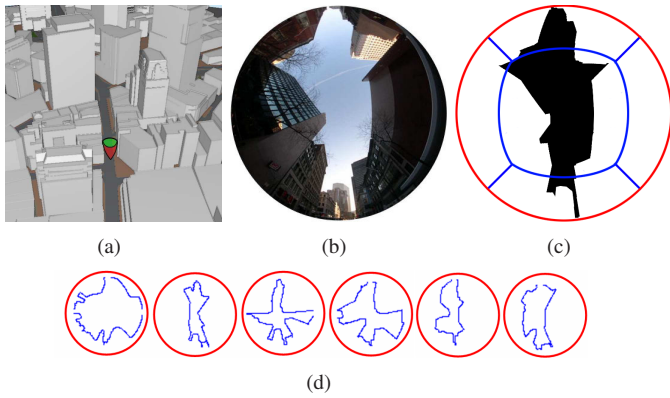


Fig. 3. Our algorithm to construct a fisheye image. (a) shows the 3D model of an urban scene where we synthesize a fisheye image. (b) The cubemap generated from a given point in the 3D model. (c) Fisheye image created from the cubemap. (d) Examples of skylines extracted from fisheye images.

B. Sky Detection

Given an omni-directional image, which is circular in the case of a fisheye model, we want to classify the pixels into sky and rest. This can be seen as a segmentation problem with two labels. The features that can be used for this segmentation can vary from simple RGB colorspace components to a wide variety of features like gradients, straight lines, vanishing points, etc. Our approach has two modules: a parameter learning method and a discrete optimization algorithm. In our problem we use graph cuts, which is both fast and highly successful in various vision problems like stereo, segmentation, and image restoration [29]. An energy function, involving binary variables in unary and pairwise terms, is represented using a weighted graph whose minimum cut (computed using the max-flow algorithm) yields an energy-minimizing partition of the variables.

We briefly introduce the energy function, whose parameters we are interested in learning. Let $x_i \in \mathbb{B} = \{0, 1\}$ where $i = 1, 2, \dots, n$, represent boolean random variables.

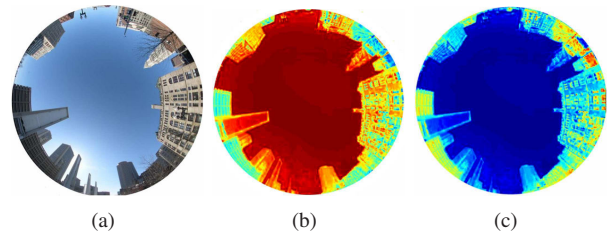


Fig. 4. (a) Original image. (b) Likelihood for the sky. (c) Likelihood for the rest of the image. Red and Blue correspond to higher and lower likelihoods respectively. [Best Viewed in Color]

In our problem, each of these variables could represent the boolean labels (sky and rest) of the pixels in the omnidirectional images. We use quadratic pseudo-boolean functions for representing our energy function. These are nothing but energy functions of boolean variables that map a boolean vector to a real value and thus the name pseudo-boolean. Let θ denote the parameters in our energy function. The parameter vector θ consists of the unary terms $\theta_{i;a}$ and the pairwise terms $\theta_{ij;ab}$, where $i, j = 1, 2, \dots, n$ and $a, b \in \mathbb{B}$. These parameters are also referred to as unary and pairwise potentials. In contrast to many vision algorithms where these parameters are manually fixed, we compute them automatically. The unary parameter $\theta_{i;a}$ can be seen as a pseudo-boolean function $f : \mathbb{B} \rightarrow \mathbb{R}$ that gives the cost when $x_i = a$. Similarly, a pairwise parameter $\theta_{ij;ab}$ is a quadratic pseudo-boolean function $f : \mathbb{B}^2 \rightarrow \mathbb{R}$ that gives the cost when $x_i = a$ and $x_j = b$. The function mapping partitions to energies is then

$$\begin{aligned}
 E(\mathbf{x}|\Theta) = & \sum_{i \in V} \{ \theta_{i;0}(1 - x_i) + \theta_{i;1}(x_i) \} + \\
 & \sum_{(i,j) \in E} \{ \theta_{ij;00}(1 - x_i)(1 - x_j) \\
 & + \theta_{ij;01}(1 - x_i)x_j + \theta_{ij;10}x_i(1 - x_j) \\
 & + \theta_{ij;11}x_ix_j \}
 \end{aligned} \tag{1}$$

where V is the set of vertices and E is the set of edges in the weighted graph $G = \{V, E\}$. Our goal is to learn the parameters (θ 's) automatically for our problem. We optimize the discriminating power of the model by estimating parameters that maximize the difference between ground truth labellings and all other labellings of a small number of manually labeled examples. Our method is similar to the maximum-margin network learning method using graph cuts [30]; we generate “near-miss” labelings, then estimate a parameter vector that maximizes the margin separating true labellings from the near misses in a convex optimization. Note that this parameter learning algorithm could also be extended to multi-label variables by encoding them using several Boolean ones [22].

In our current implementation we obtain the unary likelihood for sky and rest using their color values. We first estimate a Gaussian model for the classes sky and rest and compute the mean and covariance using manually segmented ground truth images. For a new test image we obtain the

unary likelihood by computing the Mahalanobis distance to the sky and non-sky classes as shown in Figures 4(b) and (c). In our experiments we used about 20 manually segmented images for training. We will reformulate our energy function by decomposing the unary parameters $\theta_{i;a}$ as follows:

$$\theta_{i;a} = \theta_a^p + l_i \theta_a^l \quad (2)$$

Once we have the likelihood cost l_i for every node the unary parameters θ_a^p and θ_a^l are dependent only on the label a . Similarly we assume that the pairwise parameters $\theta_{ij;ab}$ are also independent of the associated nodes i and j and replace them by θ_{ab} . Due to the problem nature, we assume that the pairwise matrix is symmetric. i.e. $\theta_{01} = \theta_{10}$. We denote the new parameter vector which we want to learn as Θ .

$$\Theta = [\theta_0^p \quad \theta_0^l \quad \theta_1^p \quad \theta_1^l \quad \theta_{00} \quad \theta_{01} \quad \theta_{11}] \quad (3)$$

The parameter vector Θ is then estimated via the standard convex program for linear SVMs, using vectors of unary and pairwise statistics from true labellings and near miss labellings as positive and negative examples. In order to guarantee that the estimated model supports optimal inference, we augment the convex program with a constraint on the pairwise terms in the parameter vector that guarantees submodularity. In the binary case the constraint is simply

$$\theta_{00} + \theta_{11} \leq 2\theta_{01} \quad (4)$$

The submodularity condition is the discrete analogues of convex functions in continuous domains.

C. Skyline Matching

First we segment the sky in fisheye images and obtain the skyline as shown in Figure 7. The predicted skylines corresponding to different locations in the 3D model are obtained using the method described in section II-A. Since the real-image segmentation and the predicted images are both high quality, a simple chamfer distance suffices to score the skyline match. During the chamfer matching, we vary the pose parameters of our virtual fisheye camera and obtain new skylines at various locations in the 3D model. The first skyline is manually matched for initialization. The subsequent ones are matched by searching for various pose parameters near the first skyline. By embedding the chamfer matching in a RANSAC framework we can handle small occlusions, e.g. due to trees as shown in Figure 5.

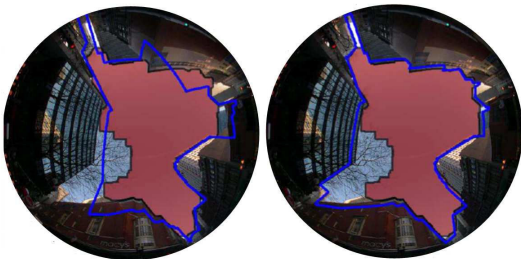


Fig. 5. Left: The skyline detected in a fisheye image with incorrect chamfer matching. Right: Robust chamfer matching using RANSAC for correct skyline matching.

D. Skyline matching using shortest path algorithm

In contrast to the earlier approach of sky detection using graph cuts and skyline matching using chamfer distance, we employ a joint strategy which proposes skylines from the 3D model and scores the matching with the omni-images. The main idea is very simple. Generate a lot of proposal skylines for various poses using the 3D model as described in section II-A. For each proposal we define a cost function which measures the similarity with the skyline in the omni-image. It is important to note that this method does not need a prior sky detection and is hence relatively more robust than the previous algorithm using graph cuts. The main steps are shown in Figure 6 and explained below:

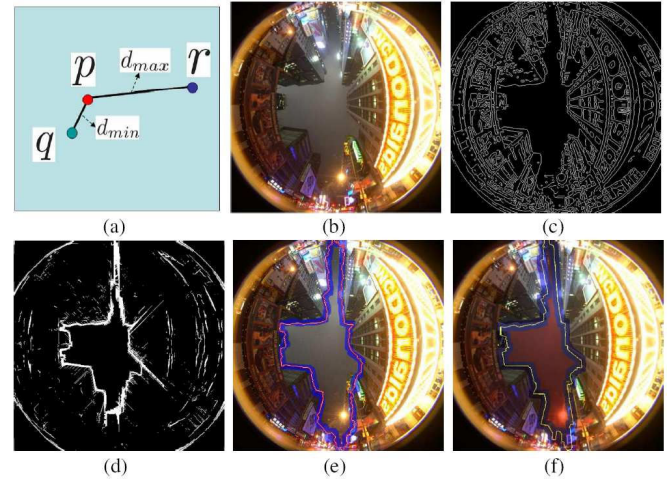


Fig. 6. (a) The max-min operator idea. (b) Example of a fisheye image. (c) Edge map on the fisheye image. (d) max-min operation on the fisheye image. (e) Shortest path result. (f) The synthetic skyline that best matches the real one.

- We detect Canny edges on the original omni-image.
- We compute the max-min operator for every pixel in the image. For every pixel p in the image, the max-min operator computes the ratio of the distance to the farthest edge pixel r w.r.t the nearest edge pixel q as shown below:

$$M(p) = \frac{d_{max}}{d_{min}} \quad (5)$$

Note that this operator highlights the edge pixels on the skylines and suppresses other unimportant edge pixels. In other words, this emphasizes the boundaries close to a large free space, i.e. sky in this case. Such free space cues have been used for alignment of 3D point clouds to overhead images by analyzing the possible camera rays from a given point [15].

- A proposal skyline is generated from the 3D model and superimposed on the omni-image.
- We dilate the proposed skyline into an area of interest with an inner boundary and an outer boundary, as shown in Figure 6.
- Next we use the *intelligent scissors* method to compute the matching cost [19]. The basic idea is to construct a graph using nodes that correspond to only the pixels

inside the two boundaries. The edges between the nodes have weights which are inversely proportional to $M(p)$. Strong max-min values induce smaller edge weights. We compute a circular shortest path inside these boundaries. The cost of the shortest path increases with the number of discontinuities. In other words, a correct proposal enables us to find a shortest path with no discontinuities. In practice, occlusions, inaccurate 3D models, discrete sampling issues in the search space, produce discontinuities even in the correct path. However, the lowest cost shortest path is always very close to the actual skyline.

The method described for skyline matching has connections to a wide variety of segmentation techniques using shortest path and graph cuts [32], [11], [27], [9]. Efficient algorithms are possible using dynamic versions of graph cuts, shortest paths or even planar graph cuts.

III. EXPERIMENTS

Our real experiments were all carried out in New York, Boston and Tokyo as shown in Figure 10 (a,b,c). In all our real experiments we used a Nikon Coolpix E8 Fisheye lens with a field of view of 183° to capture about 300 images each in New York, Boston and Tokyo. We tested our algorithm on more than 6 kms. All the 3D models were purchased from a commercial website*. We show one of the models used in our experiments in Figure 9. These are plane approximated coarse models of cities.

The day-time images taken in Boston were evaluated with the sky detection (See Figure 7) and chamfer matching algorithm. The sky detection algorithm failed for Tokyo data (due to the presence of trees) and night time images taken in New York. We used the more robust algorithm (shortest path) for the New York and Tokyo images. The geo-trajectories for the images captured in Boston, New York and Tokyo are shown in Figure 10. The images were captured approximately at an interval of 5 meters. We search for all 6 DOF for fine pose estimation using graph cuts/shortest path.

The accuracy of the algorithm is evaluated by comparing it with a commercial GPS unit "Garmin Nüvi 255W". Upward facing fisheye images and GPS measurements were collected for 30 locations in a street in the financial district in Boston. The average height of the buildings in this street is around 50 meters and the average width is 15 meters. The images were collected at the boundary between the sidewalk and the road. We register the real trajectory (boundary between road and sidewalk), GPS measurements and our localization results on an aerial image obtained from Google Earth. In order to compare the result of our algorithm the aerial map of the 3D model is registered over Google Earth by manually clicking corresponding points and by computing the transformation matrix with a least square solution. The mean error for the GPS is 29 meters and the mean error for our algorithm is 2.8 meters (cf. Figure 10). One of the reasons for the degraded performance of GPS is due to poor

satellite reception (because of tall buildings). By comparing the heights of 20 buildings in our 3D model with the ones in Google Earth a discrepancy up to 12 meters is measured. Our algorithms were able to achieve good results even with such imprecise 3D models. In the supplementary material we show our geo-localization results for various images taken in New York, Boston and Tokyo.

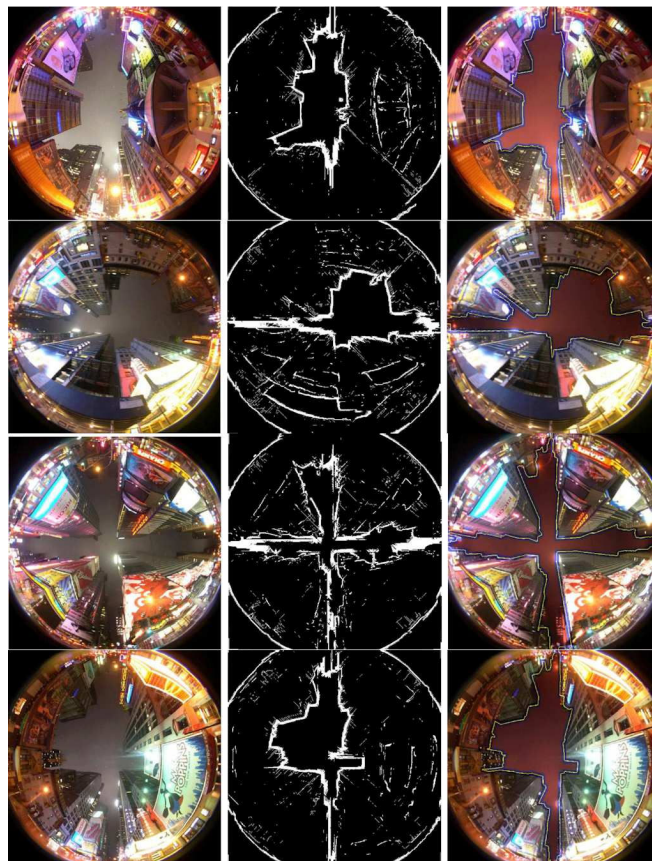


Fig. 8. On the left we show examples of fisheye images captured during night time. In the middle, we show the max-min features highlighting the boundary near an open region. On the right, we show the matched skyline with minimum cost in the shortest path algorithm.

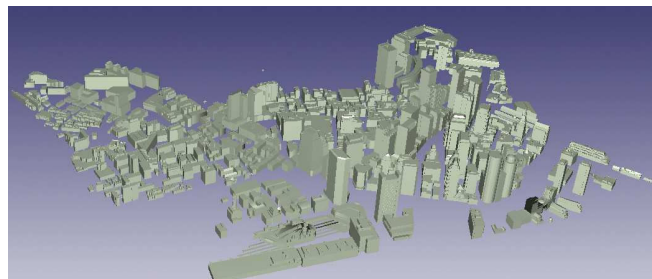


Fig. 9. 3D model of Boston's financial district: One of the coarse 3D models used in this paper.

IV. DEGENERACY ANALYSIS USING 2D-3D REGISTRATION

The skyline-based geo-localization is nothing but a 2D-3D registration using points and lines that are disguised in

*www.3dcadbrowser.com sells five 3D models for a price of 80\$ for several major cities of the world

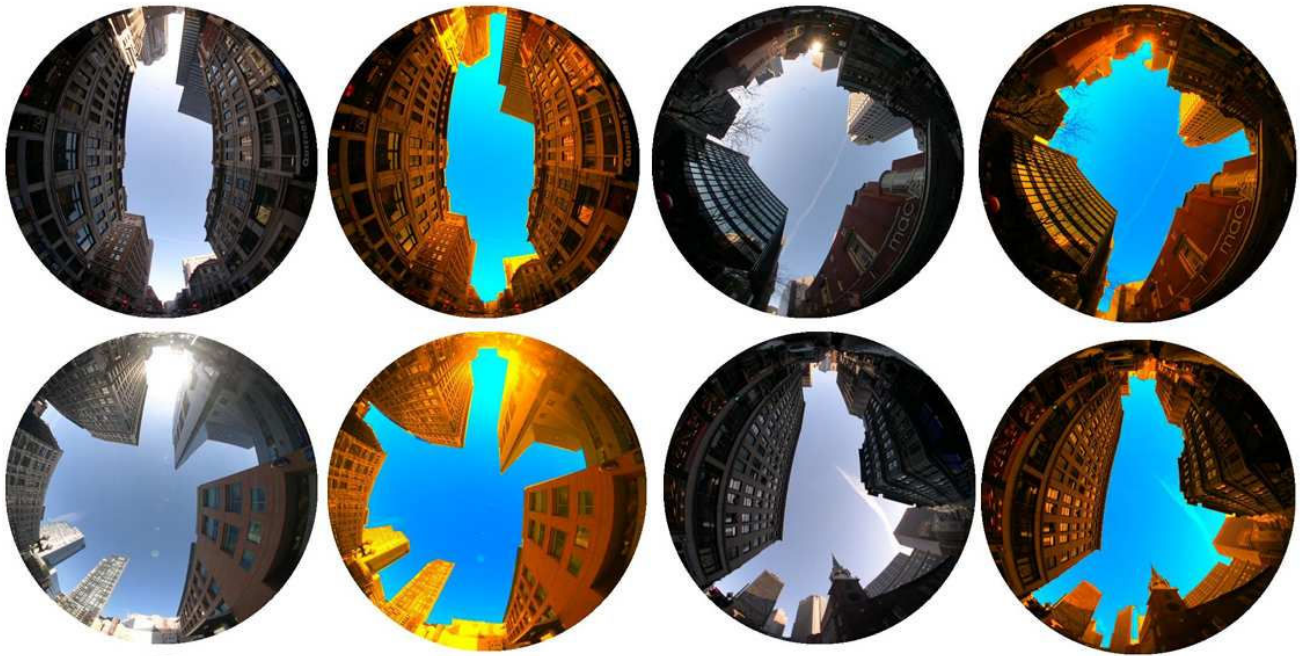


Fig. 7. Sky detection results: Original and segmented fisheye images are shown.

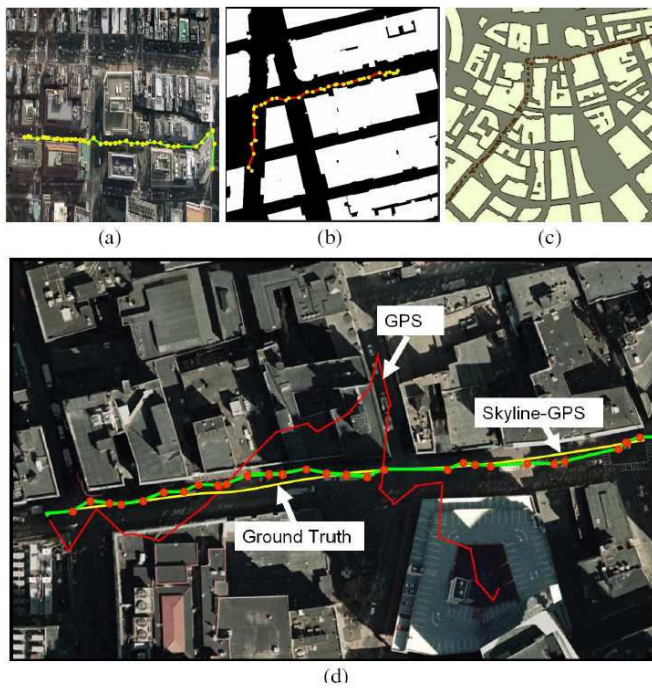


Fig. 10. On the top (a,b and c) we show the geo-trajectories obtained for 100's of images using our algorithm. In (d) we show a comparison with the GPS.

a piecewise non-linear curve in omni-images. If one realizes this connection, it becomes obvious that the theory for minimal pose estimation governs the limitations/degeneracies of the proposed algorithm. We briefly show the minimal solution for pose estimation using 2D-3D point/line correspondences. We make few reasonable assumptions in order

to find the minimal solution. First, we assume that the omni-camera is always facing upward along the Z axis. Small misalignments could be easily corrected using vanishing points [17]. Second, we assume that the images are all taken at the same height (small variations are negligible compared to the skylines). As a result there are only 3 degrees of freedom (DOF) between two consecutive images. We show a simple pose estimation algorithm that could be used to compute these 3DOF between two consecutive images. Accordingly the rotation and translation matrices are shown below:

$$R = \begin{pmatrix} R_{11} & R_{12} & 0 \\ -R_{12} & R_{11} & 0 \\ 0 & 0 & 1 \end{pmatrix} \mathbf{T} = \begin{pmatrix} T_x \\ T_y \\ 0 \end{pmatrix} \quad (6)$$

As shown in Figure 11(a), the skyline is computed for the left image using either the graph cuts or the shortest path algorithm. Our goal is to obtain a location for the skyline in the right image. Let p_1 and q_1 be two image pixels on the left image and their matching pixels in the right image are given by p_2 and q_2 . The 3D points corresponding to p_i and q_i are given by P and Q respectively. These 3D points are already known from the skyline matching in the left image. Now we have 3D-2D correspondences between the 3D model and pixels on the right image - this can be used to get the pose for the right image. Each point correspondence will give a collinearity constraint, which in turn provides two equations. We briefly explain the formulation of this constraint. As shown in Figure 11(b), the camera center for the right image C_2 , a point on the projection ray of p_2 given by $C_2 + \vec{d}_1$ and the 3D point P are all collinear. Accordingly any 3×3 subdeterminant of the following matrix vanishes.

$$\begin{pmatrix} C_{2,x} & C_{2,x} + \vec{d}_{1,x} & R_{11}P_x + R_{12}P_y + T_x \\ C_{2,y} & C_{2,y} + \vec{d}_{1,y} & -R_{12}P_x + R_{11}P_y + T_y \\ C_{2,z} & C_{2,z} + \vec{d}_{1,z} & P_z \\ 1 & 1 & 1 \end{pmatrix} \quad (7)$$

Although we obtain four equations by removing one row at a time, the total number of independent equations is only two. By using the collinearity constraint for the second point Q , we can obtain two other equations. Using these equations it is straight forward to compute the 3DOF.

Similarly it is possible to identify two coplanarity constraints from a single line correspondence. Let \mathcal{L}_1 and \mathcal{L}_2 be two projected image lines on the left and the right images respectively. Let the corresponding 3D line be given by L_1L_2 , the 3D location of it is already known using the skyline matching in the left image. As a result, the second camera center C_2 , two points on the projection rays for L_1 and L_2 given by $C_2 + \vec{d}_1$ and $C_2 + \vec{d}_2$ respectively, and two arbitrary 3D points L_1 and L_2 are all coplanar. This results in two coplanarity constraints using the quadruplets $[C_2, C_2 + \vec{d}_1, C_2 + \vec{d}_2, L_1]$ and $[C_2, C_2 + \vec{d}_1, C_2 + \vec{d}_2, L_2]$ respectively. We show the coplanarity constraint for the first quadruplet:

$$\begin{pmatrix} C_{2,x} & C_{2,x} + \vec{d}_{1,x} & C_{2,x} + \vec{d}_{2,x} & R_{11}L_{1x} + R_{12}L_{1y} + T_x \\ C_{2,y} & C_{2,y} + \vec{d}_{1,y} & C_{2,y} + \vec{d}_{2,y} & -R_{12}L_{1x} + R_{11}L_{1y} + T_y \\ C_{2,z} & C_{2,z} + \vec{d}_{1,z} & C_{2,z} + \vec{d}_{2,z} & L_{1z} \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad (8)$$

The coplanarity constraint makes the determinant of the above matrix to vanish. In the same way, we can obtain two equations from the second 3D line M_1M_2 . Overall, it is possible to get the pose from two feature correspondences (2 points, 2 lines, 1 point and 1 line).

The degenerate cases are listed below:

- Less than two distinct points on the whole skyline.
- Only one line with no distinct point on it.
- Only two parallel lines with no distinct points on them.

It is easy to observe that such cases are extremely rare to happen even in perspective images. The use of omni-images makes it almost impossible. However, the real problem is to obtain the 2D-3D correspondence, which can always be improved further as it belongs to the class of NP-hard problems. Since the degenerate cases are very rare to occur, the use of better shape matching algorithms may robustify the proposed algorithms in the following difficult conditions:

- Missing buildings - The algorithm can still work as shown in Figure 12(c).
- Inaccurate 3D models - Our experiments were all tested using coarse 3D models which were plane-based and having a height discrepancy of up to 12 meters.
- Occlusion from trees - A small amount of occlusion can be easily handled with a RANSAC based chamfer matching (See Figure 5) although the sky detection itself fails as shown in Figure 12. The shortest path algorithm can handle larger occlusions from trees.
- Repeated patterns in the skylines - The approach will fail only if the patterns repeat between two consecutive images. Note that while estimating the geo-trajectory

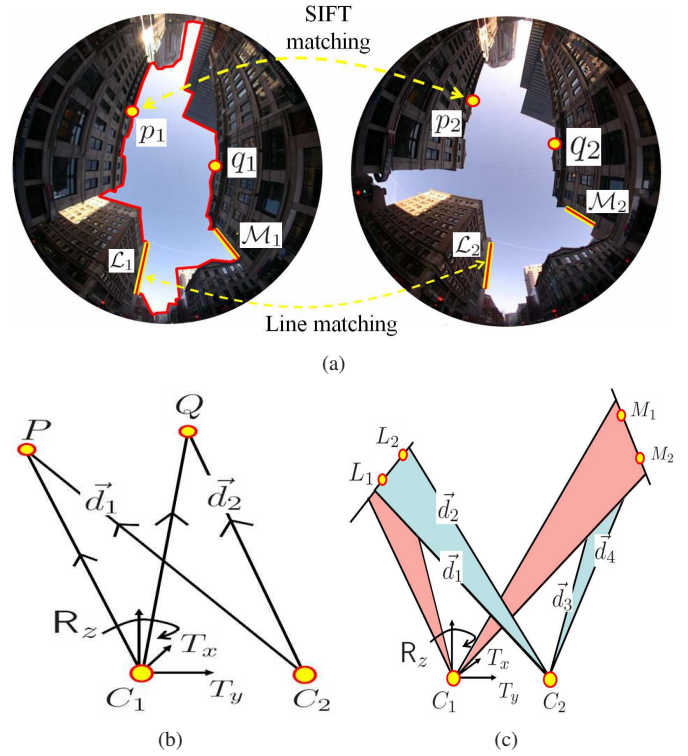


Fig. 11. In (a) we show the point and line matches between two images where the skyline is matched for the left one. In (b) we show the two camera positions for the two images shown above along with the projection rays corresponding to the two point matches. In (c) we show the projection rays corresponding to the line matches that lead to coplanarity constraints (See text).

with a good initialization, the skyline only needs to be *unique locally*.

- Sharp turns in the skylines - This sometimes degrades the performance a little as shown in Figure 12.
- Short buildings - Our preliminary experiments suggest that the method seems to work better for short buildings than tall ones because the skyline captures more lines and points.

V. DISCUSSION

Existing approaches for geo-localization use either a SLAM or recognition based techniques. This work could be seen as a method that overlaps with both these paradigms and extends to formulate the geo-localization problem as a shape matching one. This enables us to apply the existing wide variety of discrete optimization/shape matching results to this problem. Our experiments clearly demonstrate that it is possible to outperform GPS measurements in urban canyons, which are known to be extremely problematic for commercial GPS units.

The main limitation of the proposed method is that the skylines are sometimes very far away from the camera. This affects the overall accuracy of the geo-localization. However, the other advantages from skylines are numerous: easy recognition due the presence of open space or blue sky, fewer occlusions from street signs, pedestrians, cars, etc. are so overwhelming that such an approach is still very

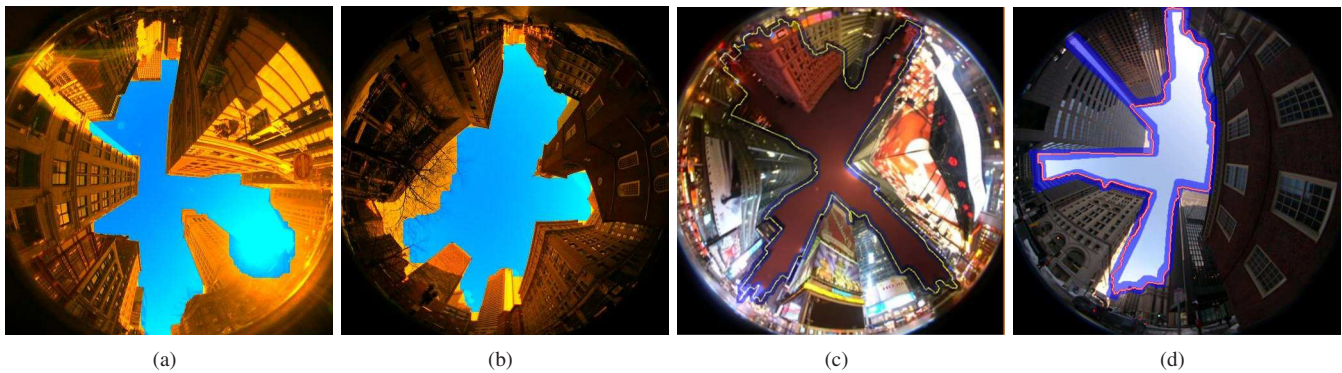


Fig. 12. A few failure cases for sky detection and shortest path algorithms. (a) and (b) show the errors in sky detection due to the occlusion from sun and trees. (c) The shortest path algorithm can handle and predict the missing buildings as shown. Mismatches in some part of the skyline while the rest matches very precisely, indicate changes in the scene. In (d) we show the short circuit problem when the boundaries for the shortest path algorithm overlap. This can result in degradation in the performance of the algorithm when the skyline takes sharp turns.

beneficial. In this work, our goal was to investigate the use of only skylines for geo-localization and we believe that it is already very good for a localization of about 2 meters. To improve the localization to a few centimeters we can use other vision features: vanishing points [17], interest point matching/tracking between consecutive frames [33], [25], 3D points reconstructed using SfM can be registered with the building walls [24], SLAM algorithms [16], [31] and maybe even priors learned from image databases [10]. It is important to note that the current GPS systems have been improved and robustified for several years by addressing various complicated issues that even include the theory of relativity.

In a few years, we believe that it will be possible to build an inexpensive, robust and accurate vision based GPS.

Acknowledgments: Srikumar Ramalingam would like to specially thank Jay Thornton for all the motivating discussions and valuable feedback throughout the project. We would also like to thank Keisuke Kojima, John Barnwell, Joseph Katz, Haruhisa Okuda, Hiroshi Kage, Kazuhiko Sumi, Ryo Kodama, and Daniel Thalmann for their valuable feedback, help and support.

REFERENCES

- [1] J.-C. Bazin, I. Kweon, C. Demonceaux, and P. Vasseur. Dynamic programming and skyline extraction in catadioptric infrared images. In *ICRA*, 2009.
- [2] T. Bonde and H. Nagel. Deriving a 3-d description of a moving rigid object from monocular tv-frame sequence. In *J.K Aggarwal and N.I. Badler, editor, Proc. Workshop on Computer Analysis of Time Varying Imagery*, 1979.
- [3] T. Brodsky, C. Fermüller, and Y. Aloimonos. Directions of motion fields are hardly ever ambiguous. In *ECCV*, 1996.
- [4] T. Broida and R. Chellappa. Estimation of object motion parameters from noisy image sequences. *PAMI*, 1986.
- [5] F. Cozman, E. Krotkov, and C. Guestrin. Outdoor visual position estimation for planetary rovers. *Autonomous Robots*, 2000.
- [6] K. Daniilidis and H. Nagel. The coupling of rotation and translation in motion estimation of planar surfaces. In *CVPR*, 1993.
- [7] A. Dunne, J. Mallon, and P. Whelan. A comparison of new generic camera calibration with the standard parametric approach. *MVA*, 2007.
- [8] H. Durrant-Whyte and T. Bailey. Simultaneous localisation and mapping (slam): Part i the essential algorithms. *Robotics and Automation Magazine*, 2006.
- [9] D. Farin and P. de With. Shortest circular paths on planar graphs. In *Information Theory in the Benelux*, 2006.
- [10] J. Hays and A. Efros. Im2gps: estimating geographic images from single images. In *CVPR*, 2008.
- [11] M. Henzinger, P. Klein, S. Rao, and S. Subramanian. Faster shortest-path algorithms for planar graphs. In *Journal of Computer and System Sciences (Selected Papers of STOC 1994)*, 1997.
- [12] D. Hoiem, A. A. Efros, and M. Hebert. Recovering surface layout from an image. *IJCV*, 75(1):151–172, 2007.
- [13] N. Jacobs, S. Satkin, N. Roman, R. Speyer, and R. Pless. Geolocating static cameras. In *ICCV*, 2007.
- [14] E. Kalogerakis, O. Vesselova, J. Hays, A. Efros, and A. Hertzmann. Image sequence geolocation with human travel priors. In *ICCV*, 2009.
- [15] R. Kaminsky, N. Snavely, S. Seitz, and R. Szeliski. Alignment of 3d point clouds to overhead images. In *IEEE Workshop on Internet Vision - CVPR*, 2009.
- [16] O. Koch and S. Teller. Wide-area egomotion estimation from known 3d structure. In *CVPR*, 2007.
- [17] J. Kosecka and W. Zhang. Video compass. In *ECCV*, 2002.
- [18] J. Meguro, T. Murata, H. Nishimura, Y. Amano, T. Hasizume, and J. Takiguchi. Development of positioning technique using omnidirectional ir camera and aerial survey data. In *Advanced Intelligent Mechatronics*, 2007.
- [19] E. Mortensen and W. Barrett. Interactive segmentation with intelligent scissors. In *Graphical Models and Image Processing*, 1998.
- [20] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 2006.
- [21] S. Ramalingam, S. Bouaziz, P. Sturm, and M. Brand. Geo-localization using skylines from omni-images. In *S3DV*, 2009.
- [22] S. Ramalingam, P. Kohli, K. Alahari, and P. Torr. Exact inference in multi-label crfs with higher order cliques. In *CVPR*, 2008.
- [23] S. Ramalingam, P. Sturm, and S. Lodha. Towards complete generic camera calibration. In *CVPR*, 2005.
- [24] S. Ramalingam, Y. Taguchi, T. Marks, and O. Tuzel. P2 π : A minimal solution for registration of 3d points to 3d planes. In *ECCV*, 2010.
- [25] D. Robertson and R. Cipolla. An image-based system for urban navigation. In *BMVC*, 2004.
- [26] E. Royer, M. Lhuillier, and M. Dhome. Monocular vision for mobile robot localization. *IJCV*, 2007.
- [27] F. Schmidt, E. Toppe, D. Cremers, and Y. Boykov. Efficient shape matching via graph cuts. In *EMMCVPR*, 2007.
- [28] F. Stein and G. Medioni. Map-based localization using the panoramic horizon. In *IEEE Transactions on Robotics and Automation*, 1995.
- [29] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. F. Tappen, and C. Rother. A comparative study of energy minimization methods for Markov random fields. In *ECCV*, volume 2, pages 16–29, 2006.
- [30] M. Szummer, P. Kohli, and D. Hoiem. Learning crfs using graph cuts. In *ECCV*, 2008.
- [31] J. Tardif, Y. Pavlidis, and K. Daniilidis. Monocular visual odometry in urban environments using an omnidirectional camera. In *IROS*, 2008.
- [32] N. Xu, R. Bansal, and N. Ahuja. Object boundary segmentation using graph cuts based active contours. In *CVPR*, 2001.
- [33] W. Zhang and J. Kosecka. Image based localization in urban environments. In *3DPVT*, 2006.