Beyond GPS: Determining the Camera Viewing Direction of A Geotagged Image

Minwoo Park
Dept. of CSE
Penn State University
mipark@cse.psu.edu

Jiebo Luo
Kodak Research Lab.
Eastman Kodak
jiebo.luo@kodak.com

Robert T. Collins, Yanxi Liu
Dept. of CSE
Penn State University
{rcollins,yanxi}@cse.psu.edu

ABSTRACT

Increasingly, geographic information is being associated with personal photos. Recent research results have shown that the additional global positioning system (GPS) information helps visual recognition for geotagged photos by providing location context. However, the current GPS data only identifies the camera location, leaving the viewing direction uncertain. To produce more precise location information, i.e. the viewing direction for geotagged photos, we utilize both Google Street View and Google Earth satellite images. Our proposed system is two-pronged: 1) visual matching between a user photo and any available street views in the vicinity determine the viewing direction, and 2) when only an overhead satellite view is available, near-orthogonal view matching between the user photo and satellite imagery computes the viewing direction. Experimental results have shown the promise of the proposed framework.

Categories and Subject Descriptors

I.4.8 [**Computing Methodologies**]: IMAGE PROCESSING AND COMPUTER VISION—*Scene Analysis*

General Terms

Algorithms, Measurement

1. INTRODUCTION

Millions of camera phones and digital cameras are sold each year world wide. With the explosion of photos and videos on the Internet, dealing with the large amount of unorganized visual data has become immensely challenging. To address this problem, one fast-emerging phenomenon in digital photography and community photo sharing is geo-tagging. The presence of geographically relevant metadata with photos and videos has opened up interesting research avenues in the multimedia research community for visual recognition of objects, scenes and events [8].

However, the current GPS data only identifies the camera location while the interesting scene in the photo may not be at the specified geo-location because it may be in the distance along an arbitrary viewing direction. Viewing direction data provided by a mobile device with a digital compass is error prone because the digital compass is sensitive to motion and magnetic disturbances.

The use of reference images has its own challenges because 1) reference images are not evenly distributed throughout the world, and 2) GPS data associated with the reference images found in the digital photo communities may be inaccurate and inconsistent.

In this paper, we address the deficiency of GPS data and scarcity of reference images by utilizing Google Street Views (covering major cities) when available and Google Earth satellite views (covering the entire globe) otherwise. Our goals are 1) to estimate the 2D viewing direction given GPS coordinates, and 2) to provide a general framework that can cover the entire world. Fig. 5 illustrates our goals with examples (with actual estimated by the proposed algorithms) taken in urban and suburban environments.

2. RELATED WORK

There is a growing body of work based on geotagged photos. Snavely et al. [13] developed the Photo Tourism system for browsing large collections of photographs in 3D. Their system takes as input large collections of images from either personal photo collections or photo sharing web sites, and automatically computes each photo's viewpoint and a sparse 3D model of the scene.

Later, Snavely et al. [12] takes a large set of community or personal photos, reconstructs camera viewpoints, and automatically computes orbits, panoramas, canonical views, and optimal paths between views.

However, these works have not dealt with mapping of data back to actual maps. To address this problem, Kaminsky et al. [5] align 3D point clouds with overhead images by computing optimal alignment using an objective function that matches 3D points to image edges while imposing free space constraints based on the visibility of points in each camera. However, their method is not suitable for estimating the viewing direction of a single arbitrary user photo because the method requires hundreds of images related to the photo.

Lalonde et al. [6] estimate camera parameters and geo-location by detecting the sun position and sky appearance from an image sequence and fitting a model of the predicted sun position and sky appearance to the detected sun position and sky appearance.

Schindler et al. [11] automatically geo-tag photographs taken in man-made environments via detection and matching of repeated patterns on building facades. Although they show very accurate results on a few image sets, their driving cue for the estimation is repeating patterns and thus the algorithm requires a database for such building facades.

Luo et al. [9] proposed a system called View Focus where they retrieve geo-tagged photos sharing similar viewing directions using bundle adjustment [7] that requires significant overlap in scene con-

MM'10, October 25–29, 2010, Firenze, Italy.

tent between community photos. However, with the exceptions of popular landmark spots for which there is a concentration of community photos, this requirement is often not satisfied in practice.

In contrast, our proposed method is a general framework that can estimate the 2D viewing direction of geotagged photos in more realistic settings. Our method only requires one input query image that is geotagged, regardless of the picture-taking environment.

3. THE PROPOSED FRAMEWORK

Our method consists of two parts, where the first part handles a case when Google Street View is available and the second part handles a case when Google Street View is not available (Fig. 1).

3.1 Part 1 - When Google Street View exists

We use *accurate* GPS information and a viewing direction associated to Google Street View. We first download 360° views of Google Street View using GPS from a user photo, collect the views related to a user photo by RANSAC-based matching, then estimate the viewing direction using our proposed method.

Download: Since Google Street View provides linked nodes where each node indicates its view center and pointers to neighboring nodes, we first identify the Street View node that is closest to the location of a given user photo and download all Street View images that are within a certain peripheral area by traversing the linked nodes. Then we generate API calls [1] that simulate viewing angle rotations at every 30° and download the simulated views. Therefore, each view contains information about both the location and viewing direction. These are the references we use to estimate the viewing direction of the user photo.

Relevant image collection: For all of the downloaded images, we follow RANSAC-based homography matching by Brown and Lowe [3]. Since rigid objects such as buildings and traffic signs are everywhere whenever Street Views are available, the RANSAC-based homography matching is effective. We collect the every image that has more than 15 matching inliers.

Viewing direction estimation: Once all the relevant images (S_i for $i = 1 \sim N$) are collected, we have N viewing directions associated with S_i and N sets of matching correspondences, M_{us} between a user photo, U, and S_i. To estimate an initial rough viewing direction, we examine each FOV (field of view) at every Street View center. We seek to find overlapping regions seen by all Street Views. Each region is given a relevance weight proportional to the number of inliers found in the previous matching (Fig. 2a). Then we use Parzen window estimation to find the highest mode of the 2D location of interesting region and obtain an initial estimate of user viewing direction as a ray from the center of user location to the highest mode (Fig. 2b).

Then we estimate a viewing direction using the fact that the point correspondences that induce a homographic mapping must be coplanar. However, we first relax the problem by considering only x-axis information from M_{us}, since the primary goal is to estimate the 2D viewing direction, which is the yaw angle. Through this relaxation, we can convert a problem of 4-degree of freedom (horizontal and vertical FOVs of the Street View, and pitch and yaw angles of the user photo) to that of 2-degree of freedom (horizontal FOV and yaw angle). Therefore, we propose the following approximation method that uses the homography constraint only and is less dependent on the accuracy of the Street View camera parameters.

We assume that the principal point of each S_i and U is the center of each image, and that the FOV of U is extracted from the camera metadata. Then we project 3D rays onto the $y = 0$ plane to form 2D rays (Fig. 3). Any plane that intersects with the $y = 0$ plane results in a 2D line on the $y = 0$ plane. Therefore, we try to find collinear points on the $y = 0$ plane. The assumption behind this is that any 3D plane that induces homography is mostly vertical, and thus the projection of that 3D plane onto the $y = 0$ plane remains to be a line, approximately.

Now we vary horizontally the FOV for each S_i and the user viewing direction to compute a hypothetical 2D point by triangulation, as shown in Fig. 3b. Since these 2D points, (x_k, y_k) for $k = 1 \sim m$ should be approximately collinear, we build a scatter matrix of the 2D points given by

$$S = \frac{1}{m} \begin{bmatrix} \sum_{k=1}^{m}(x_k - \bar{x})^2 & \sum_{k=1}^{m}(x_k - \bar{x})^2 \\ \sum_{k=1}^{m}(x_k - \bar{x})^2 & \sum_{k=1}^{m}(y_k - \bar{y})^2 \end{bmatrix} \quad (1)$$

where $\bar{x}$ and $\bar{y}$ are the mean of x_k and y_k, respectively. Then we compute the eigen values of S to measure the collinearity of the points (x_k, y_k). We select the viewing direction and horizontal FOV of S_i that minimizes a ratio $r = \frac{\lambda_{min}}{\lambda_{max}}$ where λ_{min} and λ_{max} are the minimum and maximum eigenvalues of the S.

3.2 Part 2 - When only a satellite view exists

When Google Street View is not available for an area, we download a satellite image from Google Earth according to the GPS coordinates extracted from the geotagged user photo. Since the user photo is usually a ground-level view and the satellite view is top-down from above, computing a match between them is extremely challenging because two views are near orthogonal and furthermore the appearance of common objects can vary significantly due to the different imaging conditions. That said, the ground plane and fixture objects on the ground are visible from both the aerial view and ground view (Fig. 4b and 4d). This is the basis for matching the two near orthogonal views in order to determine the camera viewing direction. Therefore detection of ground plane from a user photo and simulating ground-level view from the satellite view are important.

Ground Plane Detection: We first segment a user photo image using [4]. Then we take the boundary of each segmentation as an edge and sum all the edge responses along the x axis. This yields a vector with length equal to the height of the image. Since we do not expect the image to be perfectly normal to the ground plane, we use a box filter and convolve the computed vector with the filter. For a possible large tilt change, we can increase the size of the box filter so that we can also detect a slanted horizon. Formally, the solution is given as follows, $I_r(y) = \sum_{x=1}^{width} I_m(y, x)$ where $I_m(y, x)$ is the edge magnitude at pixel (x, y) on a segmented image and $I_r(y) = \left(I_r(y)/sum_{y=1}^{height} I_r(y)\right)$ is an edge response at vertical axis y. The solution for horizon is given as $y_{MMSE} = \sum_{y=1}^{height} y \times (I_r * BOX)(y)$ where BOX is a box filter and $*$ is a convolution operator. Since everything is a linear computation, detection takes less than a second. The plane is given by an image region confined by $1 \leq x \leq width$ and $y_{MMSE} \leq y \leq height$.

Simulating Ground-level View from a Satellite View: Since we can extract the FOV of the user camera, we can simulate a ground-level view in a certain viewing direction by rotating the FOV on the co-located satellite image, extracting image patch covered by the FOV, and warping to the ground-level view (Fig. 4c and 4d).

Alignment and Matching: We resize both images into small patches, which we call codes, to normalize the horizontal axis and vertical

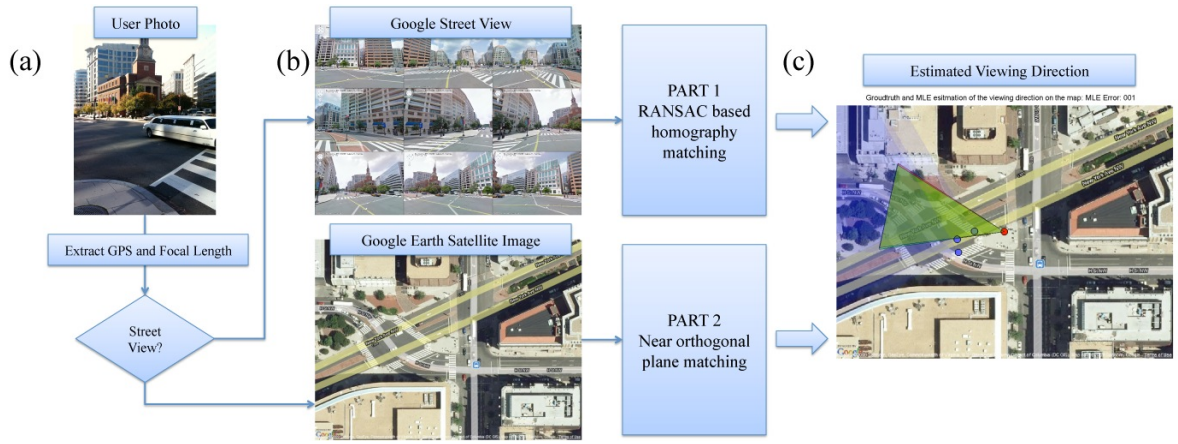

Figure 1: Overview of the proposed approach: a) GPS information tagged in a user photo is used to check the availability of reference images from the Internet. b) If Google Street View is available, surrounding views at that location are downloaded and homography-based matching is performed. If not, a satellite aerial view at the location is downloaded from Google Earth, and a novel matching between the two near-orthogonal views is performed to estimate the viewing direction. c) The estimated viewing direction is displayed on the satellite view.

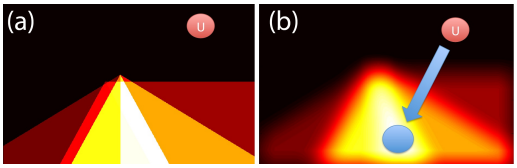

Figure 2: Initial estimate of yaw angle. The red circles in a) and b) indicate the user location and blue circle in b) indicates 2D location of interesting object. a) FOV at every Street View center. Each region covered by FOV is given a relevance weight proportional to the number of inliers found in matching. b) Parzen window estimation of interesting area seen by Street View.

axis (Fig. 4b and 4d). Since we use the same FOV when simulating a ground-level view from the satellite image, this normalization makes the horizontal axes of the two codes approximately correspond to each other. However, the y-axes that relate to distances from a camera center may not correspond to each other because we do not know the tilt angle of the camera (Fig. 4a and 4b).

If we regard the vertical axis as a time axis, there is a conceptual similarity between our matching problem and time series analysis where two signals have different speed and acceleration (e.g., speech). The similarity score of the two $3 \times w$ matrices (m_i, m_j) where w is the width of the code extracted from both codes at distance (i, j) is used to evaluate similarity between two time series at a given time (i,j) (see Fig. 4b and 4d).

Having converted our matching problem to time-series analysis, we can use normalized cross correlation (NCC) to generate a 2D disparity map between the codes and use dynamic programming to find the minimum shortest path, as can be seen in Fig. 4. Although we can use any types of appearance similarity scores and features such as the earth-mover's distance [10] and color histogram, NCC and texture help overcome the differences in terms of optics, weather, lighting and other factors originated from two extremely different imaging conditions (by a camera on a satellite vs. a consumer-level camera on the ground). Finally, we choose the viewing direction that generates the minimum shortest path as our solution.

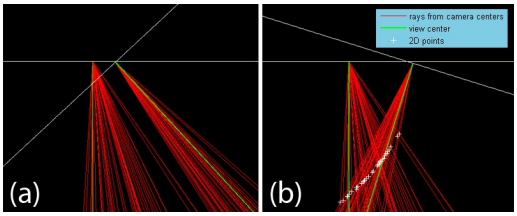

Figure 3: The red lines are rays from each camera center. The green lines are rays passing through the principal points. The white crosses are the triangulated 2D points. a) Triangulation of rays cannot find any crossing points visible by both cameras showing the given viewing direction is not possible. b) 2D points that are visible by both cameras are computed by triangulation of rays.

4. EXPERIMENTAL RESULTS

We first note the best option for accurate ground truth generation is to use traditional surveying methods that require intensive labors and expertises [2]. Instead, we have used iPhone 3GS to collect ground truth data since a manual verification of GPS and viewing direction on the spot is possible using Google Map application right on the iPhone 3GS. As a result, we built a dataset of 55 images with ground truth viewing directions in Washington D.C., New York City, Rochester, NY, and State College, PA areas for our experiments (we will make the dataset public). The dataset is small given the effort needed to obtain and verify the ground truth, but it is larger than the one used in [11]). More importantly, it covers significantly more diverse cities of various sizes and flavors (two major metropolitan cities, a mid-size city, and a college town).

Our experiments show an average mean error of 11.1° and standard deviation of 9.5° when estimating viewing direction. Fig. 5a and 5b shows examples in the urban environments using the Part 1 algorithm and Fig. 5c and 5d shows examples in the suburban or park environments using the algorithm from Part 2. Note that our algorithms can handle cases with foreground objects (Fig. 5c) as long as they do not overwhelm the scene. Moreover, Fig. 5e shows the trajectories of a person collecting data in D.C and the corresponding viewing direction estimates.

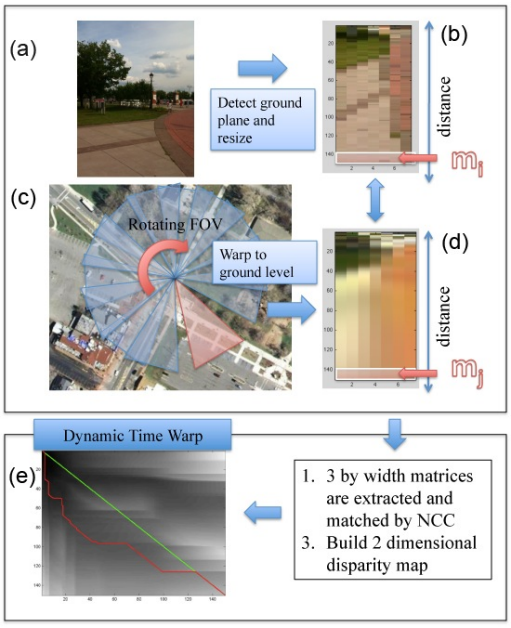

Figure 4: When only a satellite view exists: a) user photo, b) detected ground plane from the user photo using horizon detection, c) extraction of the ground plane at a specific user photo location, viewing direction, and FOV, d) simulated ground level view using the result of c), e) dynamic time warping and disparity score for b) and d).

There are failures when the structures on the ground plane are either indistinctive (looking out to the lake from the beach) or confusing (looking at two identical buildings next to each other). Remarkably, even such estimates are still roughly in the same general directions, with the worst error on the entire dataset being 37.63°.

We notice that the GPS device used for collecting the ground truth was inaccurate at the center of New York City (in the middle of the concrete jungle with maximum signal interference). This suggests a future research direction where we want to estimate both the viewing direction and (more accurate) GPS coordinates. Also the Part 2 problem is in general far more ill-posed than Part 1 and perhaps multiple co-located web photos can be helpful. We will pursue further in these directions.

5. CONCLUSIONS

We propose a general framework to estimate the camera viewing direction of a single geotagged photo in any environment and have demonstrated its promises. The main contributions are the exploitation of Google Street View and Google Earth satellite images as references, and the solutions designed to overcome various technical challenges inherent within each ill-posed scenario. Our methods perform the best when the recorded GPS coordinates are accurate. In future work, we hope to evaluate the proposed algorithms on a larger scale and further diversified dataset and refine potentially noisy GPS coordinates while estimating the associated viewing directions within the same framework.

6. REFERENCES

[1] Google APIs. http://code.google.com .
[2] Nokia Challenge (2009/2010): Where was this Photo Taken, and How? http://comminfo.rutgers.edu/conferences/mmchallenge /2010/02/10/nokia-challenge/.

Figure 5: Example results. (a),(c): user photos. (b),(d): estimated viewing directions (red triangles) using Part 1 and 2 algorithm respectively compared with ground truth (green triangles). (e) trajectories of a person collecting data in D.C and the corresponding viewing direction estimates.

[3] M. Brown and D. Lowe. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 74(1):59–73, August 2007.
[4] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.
[5] R. Kaminsky, N. Snavely, S. Seitz, and R. Szeliski. Alignment of 3D point clouds to overhead images. *Computer Vision and Pattern Recognition Workshop on Internet Vision*, 2009.
[6] J.-F. Lalonde, S. G. Narasimhan, and A. A. Efros. What do the sun and the sky tell us about the camera? *Int. J. Comput. Vision*, 88(1):24–51, 2010.
[7] M. A. Lourakis and A. Argyros. SBA: A Software Package for Generic Sparse Bundle Adjustment. *ACM Trans. Math. Software*, 36(1):1–30, 2009.
[8] J. Luo, J. Yu, D. Joshi, and W. Hao. Event recognition: viewing the world with a third eye. In *ACM Multimedia*, pages 1071–1080, 2008.
[9] Z. Luo, H. Li, J. Tang, R. Hong, and T.-S. Chua. Viewfocus: explore places of interests on google maps using photos with view direction filtering. In *ACM Multimedia*, pages 963–964, 2009.
[10] Y. Rubner, C. Tomasi, and L. J. Guibas. A Metric for Distributions with Applications to Image Databases. In *International Conference on Computer Vision*, pages 59 – 66, 1998.
[11] G. Schindler, P. Krishnamurthy, R. Lublinerman, Y. Liu, and F. Dellaert. Detecting and matching repeated patterns for automatic geo-tagging in urban environments. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.
[12] N. Snavely, R. Garg, S. M. Seitz, and R. Szeliski. Finding paths through the world's photos. *ACM Trans. Graph.*, 27(3), 2008.
[13] N. Snavely, S. M. Seitz, and R. Szeliski. Photo Tourism: Exploring Photo Collections in 3D. *ACM Trans. Graph.*, 25(3):835–846, 2006.